University of Alberta

Library Release Form

Name of Author: Bryan Ping Xian

Title of Thesis: Improving IP Multicast Over Networks

Degree: Master of Science

Year this Degree Granted: 2001

University of Alberta

Improving IP Multicast Over Networks

by

Bryan Ping Xian 

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Computing Science

Edmonton, Alberta
Fall 2001

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Improving IP Multicast Over Networks** submitted by Bryan Ping Xian in partial fulfillment of the requirements for the degree of **Master of Science**.

Abstract

In this master thesis, an overview of relevant algorithms and protocols for IP multicast is given. After an evaluation and classification of these multicast protocols, a new approach for multicast over large area network, multiple core based tree (MCBT), is proposed. The design of MCBT can greatly reduce the bottleneck of IP multicast transmission and keep a high scalability. A series of simulation on MCBT is conducted for performance evaluation and specification analysis, which is valuable for the further implementation on real world network.

Acknowledgements

I would like to thank everyone who supports me all my past two years in University of Alberta, especially my supervisor Janelle Harms, who encourage me throughout this project.

I want to thank Prof. Ursula Maydell, who just completed her academic career in Computing Science Department for more than three decades. She has given me tremendous help on simulation methodology.

At last, and for the most, I want to thank my parents who are proud of me forever. Without their love and support, I would not be who I am today.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Internet is probably one of the most amazing technologies that have been brought to this world. It connects tens of millions of computers around the world, allowing them to exchange messages and share resources. It is estimated that the size of the Internet has consistently doubled almost every year since 1969. More and more applications built on the Internet stimulate this incredible growth. Some applications that are predicted to grow fast in popularity are audio and video conferencing, video on demand, distance learning, distributed games, server and replicated database synchronization, advertising, finding and data distribution, online broadcasting: webcasting. In 1999, hundreds of thousands of viewers around the world watched Bill Clinton's video testimony on the web. Many of these applications require communication between more than two users. A serious problem behind this is the bandwidth requirement. The video or audio transmission is already a heavy load for the Internet even for one to one communications. However if the number of users is more than one, the bandwidth requirement will at least increase linearly with each additional user.

One way to solve this is to provide more bandwidth. Several new technologies, such as fiber optics, have been introduced to meet this requirement. However, it can be argued that the bandwidth requirement of applications is potentially unlimited. Considering the increasing number of users, adding bandwidth is necessary but still needs to be combined with efficient use of resources. Group communication is particularly bandwidth-intensive. Devel-

oping effective methods of delivering multi-user(multicast) services is the focus of this thesis.

1.1 What is Multicast

Multicast is the exchange of information from one or more sender(s) to multiple receivers. Here we also call *unicast* the exchange of data between one sender and one receiver. Multicast senders need to efficiently communicate with a group of receivers, without unnecessarily duplicating the data on the path to each of the receivers.

Assume that a video conference application is required to transmit a packet to 100 stations within an organization's network. It would be simple to use the existing unicast routing mechanism. However unicast transmission to the group of stations will require the periodic transmission of 100 copies of a packet and many of these packets may traverse the same link(s). Another simple mechanism is broadcast, which sends to everyone. However broadcast transmission is not an effective solution because it incurs a lot of overhead by transmitting packets to all end points even if they are not receivers. Both of these schemas require a lot of bandwidth that may be expensive and scarce. Here multicast transmission can provide an efficient delivery by requiring only a single packet transmission at the source and replication only at routers in a multicast delivery tree.

In this thesis, we use the terms *sender* or *source* to denote the origin of the information and *receiver; destination* or *group member* to denote each end-point. In terms of number of senders, multicast can be classified as *one-to-many* (one sender, multiple receivers) and *many-to-many*(multiple senders and multiple receivers). Note that for interactive (many-to-many) communication, a sender may also be a receiver. In this thesis, more emphasis has been put on one-to-many multicast, because we may regard many-to-many multicast as multiple one-to-many multicast.

1.2 Structure of Thesis

The focus of this thesis is multicast over a wide area network. In Chapter 2, IP multicast will be discussed. The basic concepts of multicast are also addressed in this chapter. In Chapter 3, a new protocol of multicast, multiple core based tree(MCBT), will be introduced to improve the multicast performance. Chapter 4 will evaluate the performance of MCBT. Chapter 5 will give the conclusions.

1.3 Thesis Contribution

This thesis explores how to provide an efficient multicast service over the Internet. All relevant algorithms and protocols for IP multicast are discussed. After the evaluation and classification of these multicast protocols, a new protocol multiple core based tree (MCBT) is proposed to improve the performance of current multicast protocols.

We implemented a simulator to investigate the performance of MCBT. The research finds that MCBT is easy to implement and has good scalability at the expense of slightly higher bandwidth cost. The protocol we propose can significantly improve the performance of IP multicast over wide area networks. A further discussion of contribution is given at the Chapter 5.

Chapter 2

IP Multicast

In this chapter, first we introduce the basic concepts of IP multicast and the operation of the Internet Group Management Protocol (IGMP). Then we describe a number of different algorithms to build multicast delivery trees that may potentially be employed by multicast protocols. After that, we focus on the multicast protocols available today. Finally we will address how to implement IP multicast on ATM.

2.1 Multicast Groups

A key concept in multicast is the notion of a multicast group (see [31]). A multicast group associates a set of senders and receivers with each other, but conceptually exists independently of them (Figure 2.1). More precisely, a multicast group is created either when a sender starts sending to the group (even if no receivers are present) or when a receiver expresses its interest in receiving packets from the group (even if no senders are currently active). The group continues to exist if at least one receiver or one sender is active. Every receiver in the group receives packets from every sender. There is no limitation on physical location or size for a multicast group. One multicast group may be distributed across the network. An individual host is free to join/leave a multicast group at any time and may be a member of more than one multicast group at any given time.

For multiple unicast, where the sender replicates packets and sends them directly to every destination, the sender must know each receiver; while a

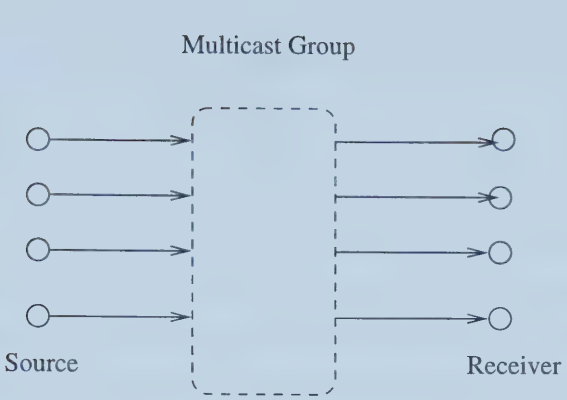


Figure 2.1: Multicast group

multicast sender can transmit packets to a group of receivers without knowing their identity. This means that a receiver can join/leave a multicast group without informing all potential senders. This simplifies implementation of multicast.

An IP multicast group is associated with a unique address. IPv4 addresses are divided into 5 classes. Multicast calls use Class D addresses. Senders send IP multicast packets with the destination field set to this address and receivers request routers to forward packets with this address to them. Local area networks (LAN) such as Ethernet have hardware-level multicast support in network interface cards. IP multicasting can take advantage of this by using a simple procedure to translate an IP multicast address to a LAN multicast address.

Two kinds of service are implemented in multicast. One service is delivering the packets between the routers in the network, which is handled by *multicast routing protocol.* The other service is delivering Packets from local routers to their attached group members, which is handled by *Internet Group Management Protocol(IGMP)*; In the next two sections, first we introduce IGMP, then we explore different approaches and algorithms to implement multicast routing.

2.2 Internet Group Management Protocol(IGMP)

Multicast packets from remote sources must be relayed by routers, which should only forward them on to the local network if there is a recipient for the multicast host group on the LAN. The Internet Group Management Protocol (IGMP) is used by IP systems (host and routers) to report their IP multicast group membership to any neighboring multicast routers. It does so by sending IGMP queries and having IP hosts report their host group memberships. IGMP is also used for other IP multicast management functions. The basic version of IGMP version 1 dates from 1988 and is now a full Internet standard. It is described in [1].

IGMP messages are encapsulated in IP datagrams. IGMP has only two kinds of packets: *Host Membership Query* and *Host Membership Report*, with the same simple fixed format containing some control information in the first word of the payload field and a class D address in the second word.

In IGMP version 1, multicast routers use IGMP to learn which groups have members on each of their attached physical networks. A multicast router periodically broadcast host membership query messages on its attached LAN to all hosts. The host will reply with a host membership report message to indicate group membership. Then the multicast router keeps a list of multicast group memberships for each attached network, and a timer for each membership. If there is more than one multicast router on a given physical network (LAN), only one of the routers will be responsible for broadcasting the queries.

IGMP version 2 [3] supports group-specific queries when the router wants to see if any hosts have left the group. When a group member sends a leave message to the multicast router, the router responds with group-specific query to this group. This new method enables quick response for group member termination. IGMP version 2 also defines a procedure for the election of the multicast querier for each LAN. The router with the lowest IP address on the LAN is elected to be the querier.

The new feature of IGMP version 3 [4] is support for *source filtering*. This new feature enables multicast group members to broadcast their interest to re-

ceive data from a specific source or all sources except one specific source. This mechanism helps conserve bandwidth when constructing multicast delivery trees.

2.3 Multicast Routing: Approaches and Algorithms

IGMP is used for forwarding multicast traffic from the local router to directly connected group members, which is the final step of multicast delivery service. It is *multicast routing* that performs the multicast delivery service from sources to routers of the receivers across the network. The goal of multicast routing is to find a tree (or trees) connecting routers who have local multicast group members and to perform multicast packet forwarding.

In this section, we will discuss how to evaluate a multicast routing algorithm. Then different routing algorithms that may potentially be employed by multicast routing protocols will be introduced. We assume that each leaf of the tree is a router that has a list of current group members, determined using IGMP.

2.3.1 Evaluation Criteria for Multicast Routing Algorithm

Generally, in evaluating a multicast routing algorithm, we may stand from the viewpoint of the multicast user and/or the multicast service provider. The multicast service users are most interested in the quality of service(QoS), which means how well the service can be provided to the user; while the multicast service providers are interested in the efficiency of the algorithm and service cost(see [7]).

The QoS criteria are considered as follows:

- End-to-End Delay: The delay can be measured as the worst case time to transmit a packet from a sender to a particular multicast group member.
- Blocking rate: For a particular multicast group, the blocking rate is

the proportion of multicast group members which can not receive the multicast packet due to lack of resources.

- Delay variation: The maximum difference between the end-to-end delays along the path from the sender to any destination nodes.
- Delay jitter: The amount of variation in the end-to-end packet transmission time from sender to one particular receiver.

Several types of service cost criteria may be considered when a multicast service provider chooses its algorithm.

- Traffic concentration: Whether or not a bottleneck(s) has occurred in the multicast tree.
- Total network cost: This may include bandwidth cost, connection setup cost and switching cost. The bandwidth cost is how much bandwidth is required to transmit the multicast data. The connection setup cost is how many control packets are used to build a multicast tree. This can also be measured as connection setup delay. The switching cost is how long for a router to switch the incoming data to the right receiver.
- Resource usage: The average and maximum link usage of links in the multicast tree.

A multicast service provider will also look at the following metrics to evaluate an algorithm.

- Scalability: Whether or not joining/leaving a multicast group is easy; whether the routing protocol maintains its efficiency when group size increases or incoming traffic grows.
- Complexity: The complexity of a multicast routing algorithm can be measured as the computation time to build the tree.

2.3.2 Approaches for Building Multicast Trees

As we have discussed before, multicast routing is the problem of finding a tree (or trees) connecting routers that have local group members. Currently there are two approaches for building multicast trees (see [2]):

- **Source-based tree**: every sender has a tree. Each source-based tree is rooted at a sender and has routes to its group members as leaves. This type of tree usually uses the shortest end-to-end delay as a factor to choose the route of the packet. The delay may be based on the number of hops or the geographical distance of the transmission path. A source-based tree is good for delay-sensitive applications, such as distributed databases and replicated files, because a source-based shortest path tree will guarantee an optimal end-to-end delay from the source to all group members. However, the scalability is poor, because a group member that joins or leaves a group may lead to the total reconstruction of all source-based trees.

- **Group-shared tree**: The construction of source-based trees shows that there will be the same number of trees as senders within a multicast group. Unlike source-based trees, only one group-shared tree is needed to implement multicast for a single multicast group. All multicast senders and receivers share this tree. This tree interconnects all group members. One host is chosen as the *center*. We can classified the centers to two kinds:

  1. *Distribution center*: this center collects the packet from the sender and then multicasts it to all group members. The group-shared tree is rooted at this center.

  2. *Administrative center*: this center manages the joining/leaving requests of group members, and advertises the status of multicast sessions. The packets are transmitted following the group-shared tree rooted at the distribution center to every group member.

In some multicast trees, one host acts as the distribution center and administrative center. While in other multicast trees, the distribution center and administrative center are different hosts.

Table 2.1(see [7]) compares the source-based tree with the group-shared tree in terms of end-to-end delay, traffic concentration and scalability. Unlike the source-based tree, a group-shared tree cannot guarantee end-to-end delay is minimal from the sender to group members. Furthermore, the group-shared tree produces higher traffic concentration because all sender and group member share the same tree, all multicast packets are transmitted via this tree, therefore bottlenecks may occur.

However, a group-shared tree has better performance in terms of scalability. If a node wants to join/leave a group, it need only send a join/leave request to the *administration center*. The center will connect/disconnect this node to the tree. Compared to a source-based tree, a group-shared tree produces lower overhead.

	End-to-end delay	Traffic concentration	Scalability	Application
Source-based tree	optimal	low	poor	delay-sensitive
Group-shared tree	not necessary optimal	high	good	frequently changed membership

Table 2.1: Source-based Tree vs. Group-shared Tree

2.3.3 Multicast Routing Algorithms

Based on these source-based tree and group-shared tree approaches, several kinds of multicast algorithms have been proposed to build multicast trees (see [31]):

1. **Flooding**

   Rule: Flooding is an algorithm for broadcast. However, many multicast algorithms are built on top of it, so we introduce this first. If the router

has not seen the packet before, it is forwarded to all interfaces except the incoming one, guaranteeing that the multicast packet reaches all routers in the internetwork. If the packet has been seen before, it is simply discarded.

Benefit: Simple to implement.

Limitation: First, every router has to store an identifier for every packet it has received, so it can identify duplicate packet. In a long-lasting multicast session, this overhead to the router memory is unacceptable. Second, flooding is inefficient, because it generates a large number of duplicate packets and uses all available paths across the internetwork instead of just a limited number. Alternately, a hop counter can be put in the header of each packet to limit the duplicate. It decrements at each hop, when it reach zero, the packet will be discarded. The initial value of the counter can be set as the hop count from sender to receiver, or if this is not available, the worst case is the two largest number of hop count between any two hosts of the network.

2. **Reverse path forwarding**

   Rule: A router forwards a packet from a source S to all its interfaces if and only if the packet arrives on the interface that corresponds to the router's shortest path to S (See [5]). To eliminate unnecessary packets of reverse path forwarding, a notion of *pruning* is introduced. Pruning is where a router informs its parent in the shortest path tree that it has no interest in receiving multicast packets from a particular source sending to a particular group.

   Benefit: Compared to flooding, reverse path forwarding does not need to check whether or not a packet has been received, this saves the router memory.

3. **Spanning trees**

   Rule: Spanning trees are built by selecting a subset of the Internet where only one active path connects any two routers which has multicast group members attached. This tree is a group-shared tree. Then the multicast

packets are transmitted via this tree.

Benefit: Avoids duplicate packets of flooding.

Limitation: A spanning tree may centralize traffic on a small number of links, and can not guarantee that the path between the source and group members is optimal.

4. **Shortest path tree**

   Rule: Shortest path tree (see [6]) is a spanning tree where the tree is constructed out of the union of shortest path routes from a source to all receivers. The concept of *shortest path* deserves some explanation. One way of measuring path length is the number of hops. Another metric is the geographic distance. However, many other metrics are also possible besides hop count and physical distance. For example, we can measure the mean queuing and transmission delay of every link so the shortest path is the fastest path.

   Benefit and Limitation: The tree structure eliminates duplicate data packet copies that would otherwise traverse those links that are common to two or more of the source-to-receiver shortest paths. The cost from the source to one receiver is optimal according to the metrics to find shortest path. However, compared to a group-shared tree, the total cost from sender to all group members may be higher.

5. **Steiner tree** *Rule*: A Steiner tree [7] is a spanning tree defined to be the minimal cost subgraph (tree) spanning a given subnet of hosts in a network. It can be used as a group-shared tree for all routers within a multicast group, irrespective of the role of sender or receiver. It scales well for large multicast groups with large numbers of senders.

   The Steiner tree problem has been studied intensively for the past half century. But it is not used in practice because:

   - Finding a Steiner tree is NP-hard, and therefore is not expected to have a polynomial time solution [8].
   - To build a Steiner tree, information about the entire network is

needed.

- Whenever a router needs to join/leave a multicast group, the Steiner tree for this group should be constructed again.

However, the Steiner tree provides the minimum link cost and serves as a good reference to measure the cost-optimality of other alternative tree types, such as shortest path tree.

2.4 IP Multicast Routing Protocols

In this section, we will introduce how the algorithms that are discussed in the last section are implemented in the following multicast protocols available today:

- Distance Vector Multicast Routing Protocol (DVMRP)
- Core Based Tree (CBT)
- Multicast OSPF (MOSPF)
- Protocol-Independent Multicast (PIM)

2.4.1 Distance Vector Multicast Routing Protocol (DVMRP)

"The Distance vector multicast routing protocol (DVMRP) computes a source-based shortest path tree rooted at each multicast sender to a group of receivers. " [9] Each router finds the best route to other routers based on the shortest end-to-end delay.

Operation

DVMRP implements the reverse path forwarding and prune algorithm that we have discussed before. The first packet from the source to the multicast group is forwarded across the entire network. The routers then send back prune messages toward the sender, if there are no group members attached to it. With this prune message, the branches from the tree that do not attach group members are removed. The router only forwards the data from its parent, if

the data is not from its parent, it will discard it. Thus, a source-based shortest path tree with all leaves having group members is created. Then every sender has a routing table to keep track of the path to every group member. After a period of time, another flooding packet is forward across the network. Then a new tree is created according to the prune message sent back to source.

Benefit and Limitation

DVMRP builds the shortest path from the sender to any group members. It guarantees that the end-to-end delay is optimal. It is most efficient with densely clustered groups of multicast receivers. However, DVMRP does not scale well for larger, sparse groups. If a host joins/leaves a group, it may change the routing tables of all senders. In order to keep the routing path up-to-date, DVMRP needs periodic flooding and pruning to compute the shortest path tree for a source. Meanwhile, each router needs to store per-source, and per-group prune records.

2.4.2 Core Based Trees (CBT)

The *Core based trees (CBT)* protocol was first addressed in [10]. The key idea in CBT is to explicitly define *core* routers that coordinate multicast. The Core is a distribution center and administration center for a CBT. The CBT protocol constructs a group-shared tree rooted at the core. The core not only administrates the multicast group, but also collects packets from sender and then forwards it to all group members.

Operation

Figure 2.2 shows how a multicast packet is forwarded across a CBT to all group members. The sender first sends the packet to the core of the particular multicast group. CBT does not require that the sender should be a member of that group. A packet from a non-group member is simply unicast toward the core until it reaches the first router that is a member of the CBT tree. When the packet reaches a member of the delivery tree, it is multicast to all outgoing

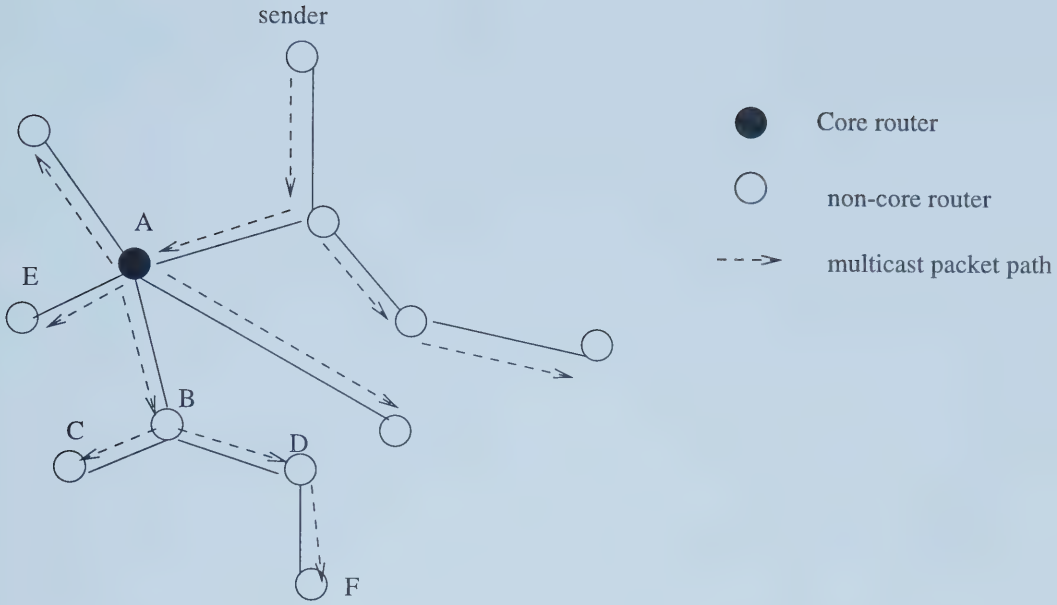


Figure 2.2: CBT Multicast Delivery Tree

branches of the tree except the incoming one. This ensures that every group member will receive one copy of the multicast packet.

The core administrates the membership of a particular multicast group. Any host that wishes to join a multicast group is required to send a *join request* toward the core. It only needs to know the address of the group core before it becomes a multicast group member. Each router along the path to the core, on seeing this join request, marks the interface on which the request arrived as the interface on which to forward packets for that group. Any host except the core that wishes to cancel the group membership is required to send a *leave request* toward the core. This group member will be removed from the CBT tree which is maintained by the core. In case when a core wants to leave, the CBT tree need to be rebuilt.

In the event of a link failure, then the downstream node of that link tries to find a new shortest path to the core. If its next hop to the core is now through one of its immediate children on the CBT tree, then this node sends a *flush message* to its children. The flush message will travel down the tree, remove the connection between all children. The children will then attempt

to reconnect along their best path to core. If the next hop to the core is not through a child, then this node will rejoin the core on a new path and does not send a flush message to its children. For example in Figure 2.2, if link AB failed, and router B found now the shortest path to A is B-D-A, then B sends a flush message to its children C and D, tearing down their connection. Finally C and D find their new shortest path to core A. If the new path from B to A is B-E-A, then B will rejoin the tree at node E and no flush message is needed.

Benefits and Limitations

Compared with DVMRP, CBT has better performance for the following measures:

- *Saving router resources:* CBT builds a group-shared tree. Thus, each router needs to store only one record per group (the interfaces on which to forward packets for that group). It does not store per-receiver information as in DVMRP.
- *Explicit join/leave:* A host can join/leave a multicast group explicitly at any time. This keeps the group membership up-to-date. While in DVMRP, a host needs to wait for the next flooded packet to update the tree for joining/leaving a group.
- *Saving network resources:* There is no bandwidth cost for the routers that are not a member of or on the path to a particular group. While in DVMRP, the flooding packet is sent to every router no matter if it has group member attached or not. When the multicast group members are allocated sparsely, CBT will save network resources.
- *Better scalability:* Since the CBT tree is a group-shared tree, the overhead for a host's joining/leaving group is much smaller than that of DVMRP.

However, CBT still has some limitations.

- *Traffic concentration:* The traffic from many sources may traverse the same set of links as it approaches the core. So the core may be a bottleneck.

- *Suboptimal end-to-to delay:* Compared to DVMRP, the path from a sender to a group member may be suboptimal. This may be a problem in some delay-sensitive applications.

- *Possible loop:* In case of a link failure, if one of this node's descendent is on its new path to core, then they will send rejoin request to each other, causing a loop. For example in Figure 2.2, if link AB failed, and router B found now the shortest path to A is B-F-A, the B and F will send join request to each other.

2.4.3 Multicast OSPF (MOSPF)

Multicast OSPF (MOSPF) [13] is built on top of *OSPF (open shortest path first)* Version 2 [14]. OSPF is a protocol specifically designed to distribute unicast topology information among routers belonging to a single *Autonomous System* . All hosts in the Internet are partitioned into Autonomous Systems. Each Autonomous System is further divided into subgroups called *Areas.* MOSF keeps a *link state protocol* database records provides a complete description of the Autonomous Systems' topology. Also, it add group membership information to this database. Thus the routers have per-group prune information. This allows a router to build a pruned tree without explicit flooding and pruning, which lead the limitations of DVMRP.

Operation

MOSPF routers maintain a current image of the network topology through the unicast OSPF link-state routing protocol. All area routers have the complete information about the topology of the area and group memberships.

- Intra-Area Routing
  If the sender and receivers are within the same Area, a shortest path tree

is constructed rooted at the sender. The multicast data is transmitted along this tree.

- Inter-Area Routing
  If the sender and receivers are within different Areas, but the same Autonomous System, a subset of the Area routers are selected to function as *Inter-Area multicast forwarders.* They are responsible for forwarding a summarized version of group membership information of their attached Areas to the backbone area. The backbone Area has complete information about group memberships in different Areas, allowing multicast data to be sent to the appropriate Areas in the Autonomous System.

- Inter-Autonomous System Routing
  If the sender and some receivers are in different Autonomous System, some Autonomous System Routers are configured as *inter-Autonomous System multicast forwarders.* MOSPF makes the assumption that each inter-Autonomous System multicast forwarder executes an inter-Autonomous System multicast routing protocol (such as DVMRP), which forwards multicast data in a reverse path forwarding manner. Each inter-Autonomous System multicast forwarder receives all multicast messages from its AS and remains on all pruned shortest-path trees.

When a host wants to join a particular multicast group, it uses IGMP to inform its local router, which then floods the network with a link state protocol packet containing this join request. After MOSPF routers receive the link state protocol packet, each (source, group) pair shortest-path tree will be rebuilt.

Benefit and Limitation

Compared with DVMRP, no flood-and-prune operation is needed for finding a multicast delivery tree. This saves network resources. Moreover, hosts can join/leave groups at any time without the flood-and-prune operation.

However, MOSPF computes a source-based multicast tree that suffers from the problems that we have discussed before. MOSPF requires a large link state protocol database, since the database must contain one record for every group

on every node in the network. Furthermore, the shortest-path computation must be done separately for every potential source, which is computationally expensive.

2.4.4 Protocol-Independent Multicast (PIM)

The motivation of *protocol-independent multicast (PIM)* is to implement different multicast routing strategies depending on whether a multicast group is *dense* or *sparse* [15]. A dense group is one where most of the routers in the network are likely to be members; while a sparse group has only a few, widely scattered members.

From what we have discussed before, we can see that for a sparse group, CBT will save lots of traffic compared to DVMRP and MOSPF. This is because the multicast packets in DVMRP and the group membership information in MOSPF are send all over the network where too many nodes are not group members. But for dense group, DVMRP and MOSPF are more efficient than CBT.

There are two versions: PIM-dense and PIM-sparse mode. Each mode exploits a unicast routing protocol to provide a routing table, but does not depend on any specific underlying unicast routing algorithm.

PIM Sparse Mode (PIM-SM)

PIM Sparse Mode is developed for sparse groups when the group members are widely dispersed and bandwidth is not plentiful. This types of groups are most common in wide-area networks. PIM-SM is similar to CBT. In PIM-SM, a core is called a *rendezvous point (RP)*. Each multicast group selects a primary rendezvous point as well as a small set of alternative backup rendezvous points. For each multicast group, there is only a single active rendezvous point. Each receiver wishing to join a multicast group sends an explicit join message to the group's primary rendezvous point. The sender to this group first sends a message to the primary rendezvous point that has registered all group members. Then the sender is replied with the shortest path to all group members.

There are two major differences between CBT and PIM-SM. The first one

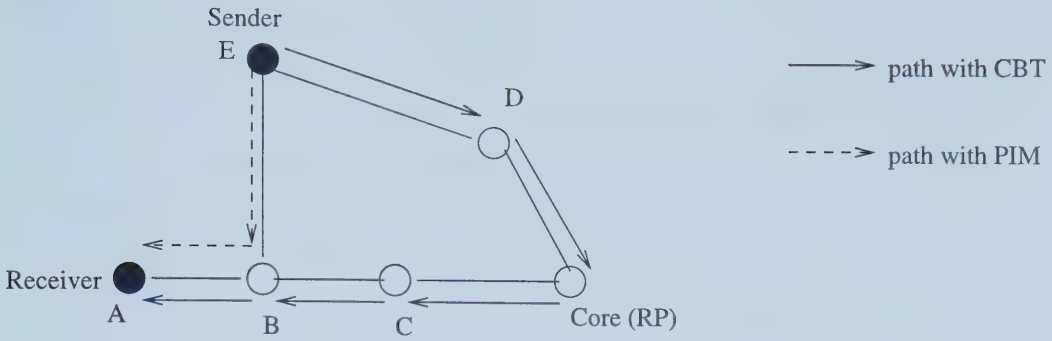


Figure 2.3: PIM-SM Vs. CBT

is that routers can choose to receive data from either the core or a shortest path tree. For example, suppose in Figure 2.3 receiver A want to receive data from sender E. In CBT, E must send data to the core first, then the core forwards it along A through C and B. But PIM-SM can choose a shorter path E-B-A.

The second difference between PIM-SM and CBT is that the rendezvous point periodically sends a message downstream advertising its existence. The leaf routers set a timer every time they receive an advertisement. If a router does not receive an advertisement in a given time and has not found a shortest path to the sender, it assumes that the primary rendezvous point is no longer reachable and sends a join message towards one of the alternative rendezvous points.

PIM dense mode (PIM-DM)

PIM-DM is developed for an environment where group membership is relatively dense and bandwidth is likely plentiful. PIM-DM is nearly identical to DVMRP. The major difference between DVMRP and PIM-DM is the introduction of explicit prune messages in PIM-DM. Unlike DVMRP, which uses flood-and-prune to periodically prune branches, PIM-DM sends explicit prune messages upstream from the leaf routers. Using PIM-DM, the router simply forwards multicast traffic downstream unless it receive a prune message.

2.5 IP Multicast over ATM

In the previous section, we have discussed various protocols for IP multicast, which are implemented in IP network. Asynchronous Transfer Mode(ATM) is another network platform that we discuss since some work on implementing IP multicast over ATM is relevant to this research. First we will explain the basic concept of ATM, in terms of its difference with Internet. Then we focus on how to implement IP multicast over ATM.

2.5.1 ATM vs. IP

The goal of ATM network is to combine the flexibility of the Internet with the per-user QoS guarantees of the telephone network. ATM is designed for high bandwidth, scalability, and manageability. There are considerable difference between ATM and the Internet that affect the implementation of multicast.

Unlike the Internet, ATM is *connection-oriented*, which means a *virtual circuit (VC)* connection must be set up before any data can be transferred between the end-points of the connection. Traffic on a VC will follow the same path and packets will be transmitted in order. Since ATM networks are connection-oriented, they can provide *QoS guarantees* by reserving resources and controlling admission. In an IP network, each IP packet is forwarded by routers on a *best effort* basis. There is no guarantee that the Internet can provide a service that can meet the requirement of every IP packet. ATM networks use fixed-sized packets called *cell*, while in IP networks, the packet length is variable. The reasons for the introduction of cells are: simpler buffer hardware, simpler line scheduling, and large parallel switches that are easier to build.

The difference between ATM and IP networks makes it difficult to implement IP multicast over ATM. The main challenge with mapping IP multicasting to ATM multicasting is the nature of the difference between IP and ATM. The ATM signaling standard 3.1 [18] supports one-to-one VCs and one-to-many VCs. Here one-to-one VC is the connection from one sender to one receiver, while one-to-many VCs are from one sender to multiple receivers.

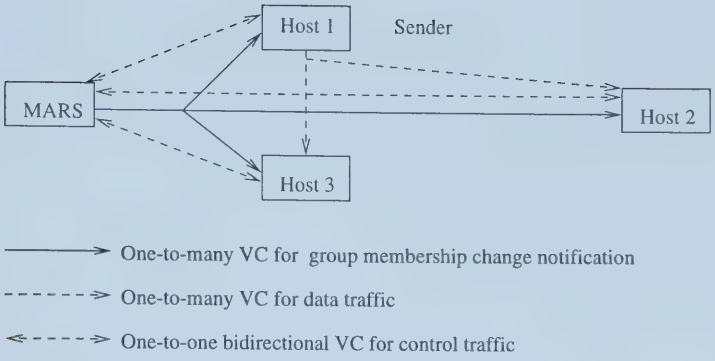


Figure 2.4: The Architecture of MARS

The connection of VC can be bidirection or one direction. There are no duplication of incoming packets along the VC, because a one-to-one connection for each (sender, receiver) pair is already built before the multicast delivery. The ATM signaling standard 3.1 also provides protocol for basic signaling function such as establishing, maintaining and releasing VCs. To implement IP multicast over ATM, we need to find a way to resolve IP addresses, manage IP groups and build ATM multicast trees on top of ATM which has limited multicast capabilities. The addressing problem will be discussed first, then two approaches for IP multicast over ATM are reviewed.

2.5.2 The Multicast Address Resolution Server (MARS)

In [21], a solution is provided to map an IP multicasting service to an ATM network. The Multicast Address Resolution Server(MARS) approach builds on the classical model of IP over ATM, emulating a shared broadcast LAN over a non-broadcast ATM network. MARS manages a cluster of ATM-attached endpoints and translates IP multicast addresses to ATM addresses. After a multicast group registers its addresses at MARS, MARS builds a one-to-one bidirectional VC from itself to each group member to carry control traffic, such as IGMP join/leave messages and address resolution requests. Meanwhile, MARS also establishes a one-to-many VC to each group member so the group member can be notified of the group membership changes. (see Figure 2.4)

2.5.3 The VC Mesh and Multicast Server(MCS) Approaches

Two approaches, VC Mesh and Multicast Server(MCS), have been proposed to implement IP over ATM on top of MARS. The VC mesh approach has been proposed in [21]. When a host wants to join a group, it has to register on MARS, which maintains a mapping of IP multicast addresses to ATM addresses. A sender which needs to transmit data to a group first sends an inquiry to MARS. MARS responds with the ATM address of every group member. Then the sender sets up a one-to-many VC that connects it to all group members. The *VC mesh* gets its name because each of the senders usually establishes a one-to-many VC to all group members, thus building a mesh of VCs for the group to communicate. The sender can add/delete outgoing VCs as it receives the update of group members from MARS.

The benefits of the VC mesh are:

- no single point of failure/congestion.
- high data throughput and optimal end-to-end delay.
- multiple senders can send at the same time.

The limitations of the VC mesh are:

- high consumption of ATM resources such as VCs and bandwidth that may be scarce or expensive.
- lack of scalability: when the host group changes, all members must initiate add/remove messages for the add/removed host, which will take a long time.
- high signaling overhead: if a sender stops sending, the one-to-many VC should be torn down.

Currently the Multicast Server (MCS) approach is addressed as an Internet-Draft in [22]. As shown in Figure 2.5, a multicast server is similar to a core of CBT. MCS retransmits data received from senders to group members. The

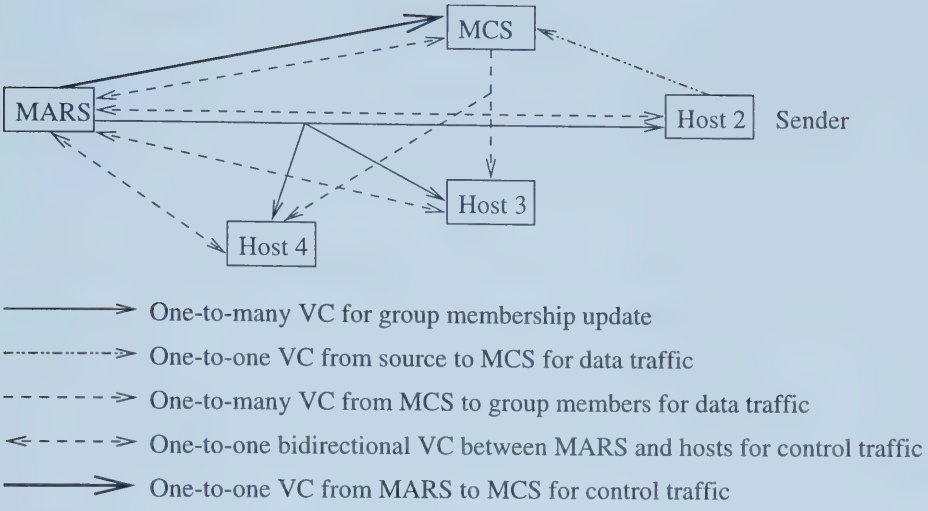


Figure 2.5: The Architecture of MCS

MCS can only serve one IP multicast group because it has no way to distinguish between traffic destined for different groups. The MCS opens a one-to-one bidirectional VC with the MARS on startup. MARS sends control traffic to MCS via this VC. Thus the MCS has registered itself on MARS. After acquiring the ATM addresses of all group members from MARS, MCS opens one-to-many VCs to all group members.

If a sender wants to transmit data to a group, it sends a request to MARS. MARS responds with the address of the MCS that supports this particular group. Then the sender opens a one-to-one VC to MCS. This one-to-one VC carries the data to the MCS. MCS retransmits the incoming data on outgoing one-to-many VCs.

The benefits of MCS approach are:

- Each host only needs one or two connection tables(Two when the host is not only a group member, but also a sender).
- MCS has good scalability: To change host group member, only the MCS needs to be informed to add or remove from the multicast tree.
- Compared to VC Mesh, MCS has lower resource consumption.

The limitations of the MCS approach are:

- The MCS may be a bottleneck and a central point of failure.
- Compared to VC Mesh, MCS has lower throughput and higher end-to-end delay, and it is difficult to handle simultaneous multiple sends to the same group.

From the discussion before we can see that the VC mesh and MCS approach both have benefits and limitations. The choice between VC mesh and MCS depends on the application on top of it. A VC mesh might be better when the group is small and stable, or the application is delay-sensitive. Meanwhile the MCS approach is a better choice when the group membership frequently changes.

In [22], a comparison has been done into the VC consumption in the MCS and VC mesh approach. Their result are shown in Table 2.2(m=the number of group members, n=the number of senders to the group). From Table 2.2,

	MCS	VC mesh
Total VCs terminated at the cluster members	n+m	n*m
one to many VCs	n+1	n
Vcs terminated at each group member	1	n
signaling requests generated due to a single membership change	1	n

Table 2.2: Cost of VC Usage in the Two Multicast Approaches

we can see that MCS can greatly save VC resources which is limited and expensive. Meanwhile, the MCS approach supports centralized control over the bandwidth usage of the multicast transmission. It is concluded in [22] that MCS is a better choice in most situations. Only when each sender has its own traffic control parameters and application is delay-sensitive does the MCS approach become less desirable than the VC mesh approach.

2.5.4 Multiple MCS Approach

In the MCS approach, all sources send data to the MCS, which forwards all incoming data to the outgoing one-to-many VCs. Thus the MCS may be a bottleneck and the failure of the MCS router may cause the failure of data

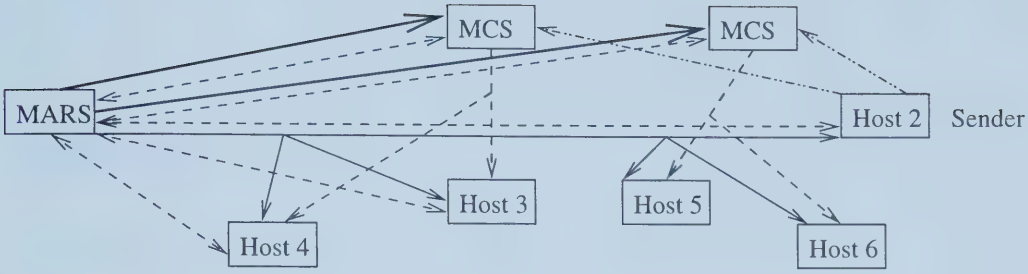
transmission to the entire group. To build a mechanism for fault tolerance and to reduce the load on the MCS, a multiple MCS approach [26] is proposed. In the next chapter, this idea of multiple cores is used for IP multicast.

In ATM, Multiple MCS use MARS to allocate existing senders/receivers to the MCSs. The receivers or senders are partitioned to reduce the load of every MCS. When partitioning receivers (see Figure 2.6), the MARS informs the MCSs of the subset of receivers they support. So every MCS will only know a subset of the receivers, and open its outgoing one-to-many VC to these receivers. The sender sends a request to MARS. MARS responds with the address of all MCSs that support this group. The sender opens a one-to-many VC to the MCSs. Every MCS forwards the incoming multicast data from sender to the subset of group members it manages via its outgoing one-to-many VC.

When partitioning senders (see Figure 2.7), every MCS only knows a distinct subset of the sender set, and opens its incoming many-to-one data VC from these senders. Meanwhile, after acquiring the ATM addresses of all group members from MARS, every MCS opens a one-to-many VCs to all group members. If a sender wants to send data to a group, it first sends data to the MCS it interconnects. Then this MCS forwards the incoming data to all group members via its one-to-many VCs.

2.6 Conclusion

This chapter has shown that many issues must be considered when selecting an IP multicast routing protocol. The network topology, QoS requirement and multicast group structure are essential for this selection. However, there is lots of room for us to improve the performance of these protocols. In the next chapter, we will introduce our proposal: Multiple Core Base Tree (MCBT).



One-to-many VC for group membership update

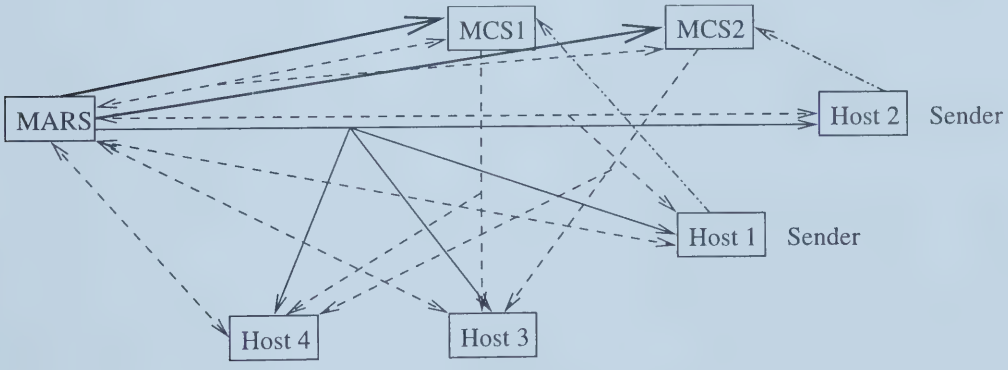
One-to-one VC from source to MCS for data traffic

One-to-many VC from MCS to group members for data traffic

One-to-one bidirectional VC between MARS and hosts for control traffic

One-to-one VC from MARS to MCS for control traffic

Figure 2.6: The Architecture of Multiple MCSs (Partition Receivers)



One-to-many VC for group membership update

One-to-one VC from source to MCS for data traffic

One-to-many VC from MCS to group members for data traffic

One-to-one bidirectional VC between MARS and hosts for control traffic

One-to-one VC from MARS to MCS for control traffic

Figure 2.7: The Architecture of Multiple MCSs (Partition Senders)

Chapter 3

Multiple Core Based Tree (MCBT): Proposal and Specification

In the previous chapter, we discussed four major multicast routing protocols, DVMRP, CBT, MOSPF and PIM. We can see that in these four protocols, CBT has better scalability and easier implementation. In this chapter, we focus our research on the Core Based Tree (CBT). Previous research on how to improve CBT performance is addressed first. Then we will introduce our strategy on improving performance by using multiple cores. This protocol is termed Multiple Core Based Tree(MCBT).

3.1 Previous Research on Improving CBT

As we can see in Section 2.4.2, the performance of CBT suffers from its traffic concentration, suboptimal-end-to-end delay and possible loops. Research has been conducted to improve CBT performance on these issues.

In [30], the researchers give a detail investigation on loops in CBT. It shows that " CBT protocol can form loops during periods of routing instability, and that it can consistently fail to build a connected multicast tree, even when the underlying routing is stable." The researchers suggest a new protocol Ordered Core Based Tree Protocol (OCBT) which maintains a logical order to reflect the level of each node in the CBT tree. Using this order, an algorithm is proposed to reconstruct a multicast tree that will keep the proper logical

orders for each node, this will eliminate loops after a link or core failure.

The choice of core of CBT influences the shape of the multicast routing tree. Thus it influences the performance of the routing scheme. A number of core choice methods have been proposed for core allocation. In [24] the authors present a brief overview of these methods.

In *Random Source-Specific Tree*, the core is chosen randomly among the sources and does not move. Doar and Leslie [25] found that "the ratio of the costs of this approach to the optimal Minimal Steiner Tree cost is typically between 1 and 2 in random graphs". The *Minimum Shortest Path Tree* calculates the actual cost for trees rooted at each group member and chooses the core as the group member that has lowest cost. The research in [7], shows that "Minimum Shortest Path Tree performs almost as well as optimal and suggests that it is adequate for use with core-based trees". In [24], the research shows that "a Minimum Shortest Path Tree has a higher delay bound than the optimal Minimal Steiner Tree". In [25], [7] and [24], the cost of a tree is defined as the bandwidth cost, which is the number of packet-hops required to deliver a packet from every source to every group member.

In [29], the author introduces three methods based on the network topology. These three methods do not require the core to be a group member. The *Maximum-Centered Tree* chooses the node with lowest maximum distance to every group member. The *Average-Centered Tree* chooses the node with lowest average distance to every group member. The *Diameter-Centered Tree* chooses the midpoint of the diameter of the group (diameter is defined as the distance between the two furthest away group members.) The research in [29] shows that compared to Random Source-Specific Tree, these three methods may reduce delay, which is defined as the maximum number of hops in the path between a source and a member. However, topology-based methods require knowledge of the whole network, which is not available sometimes.

In [12], researchers from Georgia Institute of Technology explored more of the relationship between the choice of core router and the performance of the routing scheme. Their research compared the delay and bandwidth cost performance of different core selection methods based on different routing

scenarios. Their definition of delay and bandwidth is similar to the research we addressed before.

In their research, three multicast routing scenarios are compared:

- **All Receivers Sources** Each receiver is also a source.
- **Single Source** Receivers and a single source are distributed randomly in the network.
- **Localized Receivers** All receivers are constrained to be within the same neighborhood.

Four core choice methods are studied:

- **Arbitrary** Choose the worst core from the network which has the maximum bandwidth cost or delay.
- **Random** Randomly select a core among all hosts.
- **Center** For every host of the network, calculate the depth of the shortest-path tree rooted at this host to all other hosts of the network. The topology center is the host with the minimal depth. Choose this topology center as Core.
- **Group** Randomly choose one of the receivers as Core.

	Localized Receivers	Single Source	All Receivers Sources
Arbitrary	1.9	1.5	1.6
Random	1.2	1.1	1.15
Center	1.1	1.05	1.05
Group	0.85	1.0	1.0

Table 3.1: Comparison of Bandwidth Performance for Core Choice Methods

The research in [12] is summarized in Table 3.1 and Table 3.2. These two tables present the average ratio between performance achieved with the designated core choice and with DVMRP. From these two tables we can find that the simple method of randomly choosing a core, can achieve reasonable

	Localized Receivers	Single Source	All Receivers Sources
Arbitrary	2.5	2.6	2.5
Random	1.6	1.65	1.7
Center	1.3	1.25	1.3
Group	1.15	1.6	1.65

Table 3.2: Comparison of Delay Performance for Core Choice Methods

performance. However, when group information is available, the group method will improve the performance significantly. The authors conclude that " simple methods are adequate for a widely distributed group, but that the addition of group information can be leveraged to improve performance especially when the group is small or exhibits a high degree of locality. " [12]

We can see that all research we already discussed focus on the selection of cores before the construction of multicast tree. After a tree is built, the core will remain the same. In [27], M. Donhahoo and E. Zegura suggested an algorithm to migrate the core when the group membership dynamically changes. This algorithm selects a set of hosts as candidate cores. Every time there is a group membership change, it evaluates the performance of each candidate core. If the candidate core with the best performance is significantly better than the current core, then it should migrate to this best core. The experiment in [27] shows that this new algorithm can improve the performance of CBT on bandwidth cost and end to end delay. But it can be seen intuitively that the cost of migration is high because it requires reconstruction of the whole multicast tree.

All these papers focus on how to find a single Core to improve CBT performance. But the research in [27] introduces the idea of selecting more than one core(candidate Cores) before selecting a single Core. This inspires us to investigate the use of multiple Cores to further improve performance of CBT.

3.2 MCBT: Motivation

In Section 2.3.2, we have discussed two approaches for building multicast trees: *source-based tree* and *group-shared tree.* In the previous chapter, protocols

for IP multicast and IP multicast over ATM which are based on these two approaches are introduced. Table 3.3 classifies these protocols into two categories. Table 2.1 shows that source-based trees lack scalability but have low traffic concentration; while group-shared trees have good scalability but high traffic concentration. Currently many kinds of web-based multicast applica-

	source-based tree	group-shared tree
IP multicast	DVMRP, PIM-DM	CBT, PIM-SM
IP multicast over ATM	VC mesh	MCS

Table 3.3: Classification of Multicast Protocols

tion, such as videoconference and distributed games need to support frequently changing group membership, high traffic load and large multicast group size. Thus high scalability is essential for them. However, the existing group-shared protocols, such as CBT or PIM-SM for IP multicast and MCS for IP multicast over ATM, may cause high traffic concentration at the center (core or MCS). A new approach is required to reduce the traffic concentration without losing scalability.

Here we introduce the *multiple core based tree (MCBT)* strategy to meet this challenge. MCBT builds a group-based tree where multiple cores are selected within a group. Unlike CBT, where only one core acts as a *distribution center*, MCBT has multiple cores to collect the incoming packets and then multicast them to their group members. This mechanism reduces the traffic load of a single core without sacrificing scalability.

In the design of the MCBT protocol, besides high scalability and low traffic concentration, other goals need to be achieved:

- It is important to keep the senders and the receivers(group members) ignorant of the allocation of each other. This will reduce the storage of senders and receivers.

- One group member only receives multicast data from an associated core. That is, it should not receive duplicate copies of multicast data from other cores.

- It will be desirable if the selection method of cores is straight forward. This will reduce the setup cost of the multicast tree.

To select multiple cores, there are two choices: *sender-based and receiver-based.* For sender-based, a particular core serves a subset of senders, and maintains a multicast tree to all group members. The sender first sends data to its associated core, then this core multicasts incoming data to all group members. For receiver-based, each core serves a subset of group members. The multicast tree rooted at this core reaches this subset of group members. The sender sends data to all cores within that group, then all cores forward the incoming data to the interconnecting group members.

MCBT is designed for the Internet, which has no central control. Anyone can be a sender to a particular group, therefore information of all senders is not always available. Thus it is difficult to maintain a sender-based multiple core based tree. In our research, we assume that we always know the group membership before any multicast service begin. This is because before a multicast application (webcasting, Internet pay-per-view etc) happens, an administration center should know the group members, then this center can know the service cost, and let the multicast user know the service charge. Therefore we choose to have MCBT partition receivers instead of senders.

3.3 The MCBT Approach

MCBT tries to improve the performance of CBT with multiple cores and partitioning receivers. In this section, the approach of MCBT will be discussed. The structure of MCBT will be explained first. Then we will explore the role of cores (distribution center) and root (administration center). The mechanism of core selection, group management, building a multicast subgroup and MCBT operation will be addressed accordingly.

3.3.1 MCBT Structure

An example of MCBT is shown in Figure 3.1. The multicast group is divided into two subgroups. Each group has its own core. The sender sends the

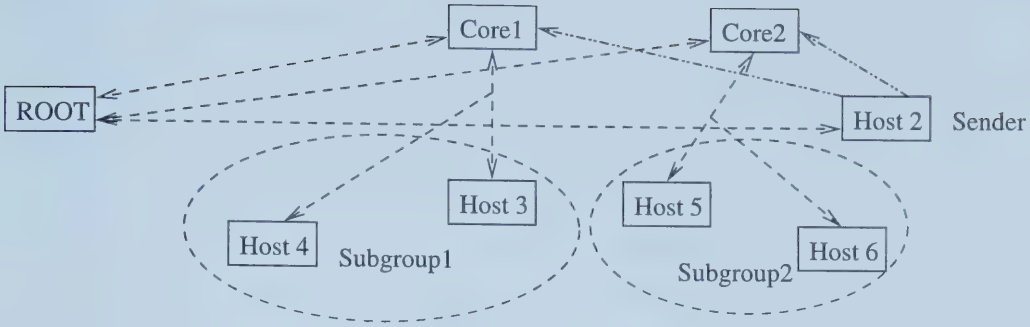


Figure 3.1: Structure of MCBT

multicast data to these two cores; then the cores multicast this incoming data to their associate subgroup members. In our research, we investigate the MCBT where the number of cores in one multicast group is 2. However this number may be more than two. The choice of number of cores will be discussed later.

3.3.2 Core & Root

There are two kinds of centers within one group: core and root. A multicast group has only one root; while it may have multiple cores. The root is responsible for core management. It is the administration center of the cores. In the Internet, the router with maximum resources (bandwidth, processing power etc) may be chosen as root. The root keeps the allocation of cores when the multicast group is constructed. MCBT tries to reduce the load of the core with the partition of group members. Every subgroup will have its own core. The core maintains a multicast tree to its subgroup members. After a group member is chosen as a core, it registers itself to the root. When a sender first wants to send data to a multicast group, the sender will ask the root for the core addresses. A core is a distribution and administration center (see Section 2.3.2) to its subgroup. It not only manages the joining/leaving request of its subgroup members, but also receives the packet from the sender and then retransmits it to all subgroup members. The core maintains a group table that records all paths to its group members.

3.3.3 Core Selection

In Section 3.1, previous research on how to improve CBT performance has been explored. The investigation conducted in [23] presents some valuable conclusions for the single core choice, which is applicable for the core choice method in MCBT. "Simple methods are adequate for widely distributed groups, but the addition of group information can be leveraged to improve performance especially when the group is small or exhibits a high degree of locality. "

As we defined in Section 3.2, an administration center knows all the group members and all the group members know this administration center before a multicast application happens. In MCBT the root is the administration center for all cores. If the root does not know the allocation information of multicast group members, such as the topology location of each group member, then the root may randomly select two existing group members as cores. If the root knows more information of the group members other than the group membership, such as topology location of each group member and link state information (transmission rate, delay, available bandwidth). Then any node can figure out distances to other nodes. Then the root can find two cores by using this information.

In our simulation, we implement two core choice methods:

- **Randomly Distributed Cores.**

  The core of each subgroup is randomly chosen from the multicast group members. We call this MCBT scenario as MCBT(random). Suppose we want to partition every group into two subgroups. So first we randomly choose a group member as root. Then two group members at random are chosen as cores. This method is analogous to the group method conducted in [23], which we have introduced in Section 3.1.

- **Sparsely Distributed Cores.**

  Although MCBT(random) is easy to implement, it may suffer some problems. To reduce traffic concentration, we want the load of the two cores to be balanced. As we assumed, every host who wants to join the multicast group should inform the root. Besides the membership, if the root

knows the allocation and link state information of all group members, then the root may use the information to choose cores. Suppose all group members are evenly allocated in the network, we may split the network into two areas. All the multicast group members in the same areas are assigned into one subgroup. Instead of randomly selecting two group members as cores, we may choose two group members that are distributed as sparsely as possible. Then the maximum geographical distance from core to its subgroup members will be reduced. We call this scenario as MCBT(sparse). If the group memberships keep changing, then a dynamic schema of core selection is needed to keep the cores sparse. The root may periodically re-elect two sparse cores according to the new group allocation.

3.3.4 Building Multicast Subgroup

After some group members are designated as cores by the root, other group members send join request to the root. The root will respond with the addresses of the nearest core for each group member. Then this group member joins this subgroup. Here, the concept *nearest* deserves some explanation. In our design of MCBT, we use the geographical distance as the measurement of path length. After the core receives the join message from other group members, a shortest path tree originating at the core is constructed to the receivers.

3.3.5 Group Management

The group management of MCBT is quite similar to that of CBT. If a host wants to join a multicast group, first it sends a join request to the root. The root chooses the nearest core for this host, then the root registers this host to the associated core. The core adds the shortest-path to this host in its group table.

If a host wants to leave a multicast group, it sends a leave request to its core. Then the cores update its group tables. Because we choose a group member as core, the core may also leave that group. If this happens, the root will have two

choices, one is to choose another group member within that subgroup as core, then rebuild a new multicast tree in that subgroup. Or the root reselects two new cores using the core selection method that we suggest, then reconstructs two new multicast trees. Comparing these two choices, the first choice will have less tree reconstruction cost but all group members may not connected with the nearest cores and cores may not be sparse if we use Sparse method. Meanwhile, the second choice can ensure the group members are connected to the nearest core and the cores are sparse, but the reconstruction cost is higher. In the implementation, we suggest the second choice can be used if the group size is small. If the group size is big, to save the reconstruction cost, the first choice is better.

In the event of a link or node failure, the operation of MCBT is the same as that in CBT (See Section 2.4.2).

3.3.6 Operation

Before a multicast application happens, we assume that the root knows all the group members. Then the root constructs multicast trees based on the core choice method we discussed in Section 3.3.3. After these multicast trees are built, if a host wants to send data to a multicast group, first it sends a request to the root of this particular group. The root will respond with the addresses of multiple cores. Then the source builds a multicast tree from itself to all cores. After the construction of the multicast tree from source to cores, the data will be sent to the cores first. Then the core will forward all incoming data to all group members through the group table it maintains.

3.4 Benefit and Limitation

The design of MCBT has benefits of:

- **High Scalability**
  The design of MCBT inherits its scalability from CBT. In MCBT, usually the joining/leaving will not lead to the reconstruction of all multicast trees. After receiving a joining/leaving request, the core only needs to

add/prune a multicast branch as in CBT. Thus the joining/leaving of a group member will only affect the tree of one subgroup unless this group member is a core of this subgroup. We will investigate the ability of MCBT to handle high incoming traffic and large multicast group size in the next chapter using simulation.

- **Low Traffic Concentration**
  In CBT, the core receives the traffic from all senders to this group and forwards to all group members. The core may be a bottleneck. In MCBT, the traffic load has been partitioned. A core only needs to forward the incoming traffic to the subgroup members it serves. The root is not involved in the distribution data, so unlikely to be a bottleneck.

- **Easy to Implement**
  In MCBT, the receivers don't have to know the allocation of the senders. The senders and receivers communicate via the root. The root informs the sender of the address of every core after it receives a request from the sender. Thus the senders do not need to know how to construct multicast trees, they only need to find the path to cores and send the data to cores.

However, there may be some limitation. We can imagine that with more multicast tree handle the incoming traffic, the connection setup cost will be higher than that of CBT. Also the bandwidth cost for MCBT would be higher than that of CBT because multiple trees can not merge the bandwidth as well as a single tree. We will investigate this in our simulation.

3.5 Applicability On Large Area Network

MCBT can be used to provide multiple cores for multicast in a local domain. It can also scale well for a large area network. If we consider an internetwork, it has a backbone with a connected group of routers from different domains. Each router on the backbone manages the hosts in its domain. Suppose the group members are distributed widely across a wide area, then for each domain that

contains multicast group member, a router of this domain which also on the backbone can be assigned as the *local core* of this domain. A local core based tree can be built for the multicast group members within this domain. This is useful since the group members within the same domain are likely close to each other. By grouping them together, the overhead of group management and data transmission will be reduced. After the construction of local core based trees, we may select multiple *primary cores* in the backbone. The core selection method discussed in this thesis can be used to find the primary cores. Routers who have local core based trees in their domain can join the nearest primary core. Thus a hierarchy of cores is constructed. In this hierarchy, at the top is a multicast tree from the sender to all primary cores. The next level is the multiple core based trees rooted at primary cores, where the leaves of these trees are the local cores from different domain. At the bottom, every domain has a core based tree rooted at the local core, where leaves are multicast group members within this domain.

3.6 MCBT vs. Multiple MCS Approach

It would be interesting to compare MCBT and the multiple MCS approach, since the inspiration of MCBT comes from multiple MCS. We can see the function of *root* in MCBT is quite similar to MARS; while cores act as MCS. However, there are still some differences between MCBT and MCS because of the nature of IP and ATM.

The multiple MCS approach builds connections from MCS to receivers with one-to-many VCs. We can simplify this connection as a multicast tree rooted at MCS. In this tree no nodes are multicast-capable, which means they can not duplicate an incoming packet. The motivation behind this is to reduce the overhead of interconnection routers. However this is at the cost of high bandwidth because of the duplication of incoming packet.

The notion *multicast-capable* deserves some explanation. If a host is multicast-capable, it can duplicate an incoming packet. Suppose in Figure 3.2, all hosts are multicast-capable. If r want to send a packet to m and n, it only needs to

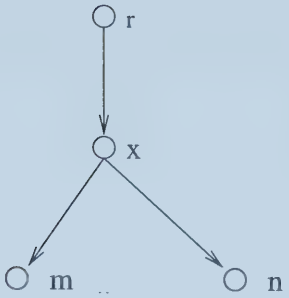


Figure 3.2: Multicast-capable

send to X once. X will duplicate this packet, and then forward it to m and n. If X is not multicast-capable, then r needs to send this packet twice to X. MCBT exploits the Internet that may have some multicast-capable nodes.

3.7 Conclusion

In this chapter, we introduce our strategy to improve multicast performance: Multiple Core Based Tree(MCBT). The design of MCBT tries to keep high scalability and low traffic concentration. In the next chapter, we will do a series of simulation experiments to study the performance of MCBT in a wide area network.

Chapter 4

Simulation Study

The motivation of MCBT is to improve the performance of CBT on traffic concentration without losing its scalability. A series of simulation experiments are designed to compare the performance of MCBT and CBT. The experiments simulate an internetwork with multiple senders sending multicast data to a multicast group. The performance of MCBT and CBT based on bandwidth cost, delay, traffic concentration and blocking is investigated.

4.1 Simulation Overview

To evaluate the performance of MCBT, we compare it with CBT. We build our simulator with C language. This simulator tries to simulate when there are a number of multicast call to the multicast group, the performance of the each algorithm.

As discussed in Section 3.1, the performance of CBT greatly depends on the core selection method. To compare CBT and MCBT fairly, we try to use similar core selection methods.

First, we investigate their performance when the cores of CBT and MCBT are randomly chosen:

- **CBT: Randomly Chosen Core**
  In Section 3.1, we have introduced the investigation conducted by E. Zegura and M. Donahoo (see [23]). Their research found that simple methods are adequate for a widely distributed group. As shown in Table

3.1 and Table 3.2, the *Group Selection method* achieves good performance. So we randomly choose a group member as core.

- **MCBT: Randomly Chosen Cores**
  For MCBT, we must partition every group into two subgroups. After a group member is randomly selected as root, two group members are randomly chosen as cores. The group is partitioned by the distance to each core.

The performance of the schema that explores more group information is also investigated. In this case, the cores are placed to hopefully provide more balanced load.

- **CBT: Diameter-Centered Tree**
  If all member topology location are known in advance, we may implement *Diameter-Centered Tree* to improve the performance of CBT. This scenario for CBT is called CBT(DCT) (see [29]). The core is the midpoint of the diameter of the group, which is defined as the distance between the two furthest away group members.

- **MCBT: Sparsely Chosen Cores**
  In this simulation, one multicast group is divided in 2 subgroups. Inspired by CBT(DCT), a method is proposed to implement MCBT(sparse): First, a group member is randomly chosen as root. We select a pair of receivers who are maximum geographical distance apart. This method ensures that the selected two cores are located as far apart as possible within the graph. The hope is that the subgroups will have more balanced load.

The performance of CBT and MCBT are compared for both scenarios.

4.2 Terminology

In the simulation, the following terms are used:

- **Call**: a call is the process where one sender sends data to a group.

- **Event**: two kinds of events are in this simulation: call arrival and call departure.

- **Link**: In this simulation, we call the undirected connection between two hosts a *link*. Every link has attributes of *geographical distance, bandwidth capacity and used bandwidth.*

4.3 Efficiency and Performance Measures

In order to compare the performance of CBT and MCBT, five types of measurement are considered: *delay, bandwidth cost, link utilization, core concentration and blocking.* They can represent the QoS and service cost of MCBT and CBT (see Section 2.3.1). From the investigation of these performances, we can find the scalability of these two algorithms.

The performance measures are defined as follows:

- **Delay**: For every call, delay is the geographical distance on the path between the source and the most distant group member. In this simulation, we measure the average delay over all calls. Suppose D_i is the delay of call i from the sender to the furthest group member, and N is the total number of calls handled in the simulation, then delay is $\sum_{i=1}^{N} D_i/N$

- **Bandwidth cost**: The bandwidth cost is the average number of bandwidth-hops required to complete a call. The cost for one call is the number of hops in the shortest path from sender to core(or cores) plus the number of hops traversed to send a packet from core(or cores) to all group members, then multiplied over the required bandwidth of this call. We sum the cost for all calls, then get the average cost per call as the bandwidth cost. In CBT, suppose all hosts are multicast-capable, then the required number of hops to send a packet from core to all group members is the number of hops on the shortest path tree rooted at the core. If not all hosts are multicast-capable, then this is different. For example in Figure 3.2, a call is forwarded from r to both m and n via node x. If all hosts are multicast-capable, the hop cost is 3. If x is not multicast-capable,

then the hop cost is 4, because the same message must be transmitted twice over the link from r to x: once addressed to m and once to n. We can see that the bandwidth cost will increase if not all hosts are multicast-capable.

- **Link utilization**: Suppose we have N incoming calls. Then there will be N call arrivals and N call departures. Every time there is a call arrival and call departure, the link usage will change. We call this link usage, the link state. We calculate the average link utilization of link k based on the its weighted duration of each link state. D_j is th duration time of for every link usage $j, j = 1..2N-1$; $UB_{k,j}$ is used bandwidth of link k during this link usage $j, j = 1..2N-1$; LB_k is the bandwidth capacity of link k; LU_k is the link utilization of link k and T is the simulation duration time from the first call arrival until the last call arrival.
$LU_k = \frac{\sum_{j=1}^{2N-1} \frac{UB_{k,j}}{LB_k} D_j}{T}$
We measure the average link utilization over all links in the network.

- **Core concentration**: In the multicast tree, there are active links originating from the core. An active link means there is traffic of this multicast group on that link. Core concentration is the average of the link utilization on these active links. In CBT, we investigate the core concentration of a single core. In MCBT, we have 2 cores within one group and the core concentration of these two cores are measured separately.

- **Blocking**: For every call, the blocking is the ratio of the sum of blocked receivers to the sum of possible receivers. Blocking occurs when there is not enough bandwidth on the path from the core to a group member to transmit incoming data. Then this group member can not receive this data. Blocking also happens if the bandwidth from the sender to core is not enough. In this case, all group members of this core are blocked. In this simulation, we count the average blocking rate over all calls.

4.4 Simulation Parameters

There are many parameters in this simulation corresponding to network, multicast group and call characteristics.

4.4.1 Network Parameters

We model a network as an undirected graph, in which nodes represent routers and edges represent links between routers, using a random graph model suggested by researchers at Georgia Institute of Technology in [23]. Using this model a flat graph is generated to represent a backbone connecting different domains of the Internet. Each node in this graph represents the router of a domain.

The parameters used to model the network are:

- **number of hosts**: 100 nodes are generated for our experiments.
- **link**: the properties of link are *physical distance and bandwidth capacity.* The physical distance is generated with the graph. In this simulation, we assume every link has the same bandwidth capacity of 4 Gbps.

4.4.2 Call Parameters

Every call has its *required bandwidth, inter-arrival time, duration time, sender host and receiver group.* This simulation supposes the call duration time and call inter-arrival time are exponentially distributed.

- **required bandwidth per call**: This is a factor in certain experiments. In other experiments, the required bandwidth cost per call is fixed to 50kbps.
- **call duration time**: this is exponentially distributed, with the mean call duration time as 30 second.
- **call inter-arrival time**: this is exponentially distributed, with the mean call inter-arrival time as 20 second.

The length of the simulation is given by the *Number of calls*. We will determine this in Section 4.6.2.

4.4.3 Multicast Group Parameters

This simulation models a videoconference application, where every host is a possible source and may send data to any group. The sender is not required to be a group member. In our simulation, we assume that the group membership is always known before any multicast service begin. The multicast groups are assumed to be static, so no joining or leaving for any receivers.

The multicast group parameters are:

- **group size**

  An experiment will be conducted on various group sizes when we investigate the performance of CBT and MCBT. In other experiments, 30 nodes are randomly chosen as multicast group members.

- **Multicast-capable**

  In this simulation, each node is designated as either multicast-capable or not. An experiment is used to see how sensitive the results are to the degree of multicast-capability on CBT. The same incoming traffic, group member, core and multicast tree are created; while the factor of multicast-capable nodes is increased for each run. To see the effect of multicast capability, some simulation results are shown in Table 4.1 and Table 4.2. We can see that the core concentration and blocking rate will increase significantly with the decrease of multicast-capable rate. This is because more duplicated multicast data on the tree will cost more bandwidth, and lead to more core concentration and bottlenecks.

multicast-capable proportion	0.25	0.40	0.50	0.75
core concentration	0.82	0.40	0.29	0.15

Table 4.1: Comparison of Core Concentration on different multicast-capable rate

multicast-capable proportion	0.25	0.40	0.50	0.75
blocking rate	0.20	0.11	0.05	0

Table 4.2: Comparison of blocking rate on different multicast-capable rate

In the rest of our experiments, we randomly designated 50% of the hosts as multicast-capable. From our experimental results we can see this will provide a reasonable performance. Also, this proportion has been used in other research papers (see [12]).

- **Number of Groups**

  We can imagine if there are more than one multicast group within a network, then the multicast trees for different multicast groups may share some branches. These shared branches may be potential bottlenecks because they have traffic from different groups. Then the blocking rate may increase. This will happen to both CBT and MCBT. The other performance measures (bandwidth cost, delay and core concentration) depend on the structure of multicast tree and not the number of groups. Thus the number of groups will not have significant impact on the comparison between CBT and MCBT. In our simulation, we set the number of groups to 1.

4.5 Simulation Process

There are four basic steps in the simulation experiments. These steps are put together in different flow depends on the experiment factor.

1. **Building of the Multicast Group**

   After the input of graph model and simulation parameters, a number of nodes given by the group size parameters are randomly chosen as the group. Meanwhile, according to the simulation parameters, we assign the link capacity of every link to 4Gbps and randomly designate every node as multicast-capable or not.

2. **Building of the Multicast Tree**

   Based on the different multicast scenarios, cores are selected for CBT and MCBT. Then the shortest path tree from the core to every group (subgroup) members is constructed. In this simulation we consider the link length, which is the sum of delays on the links of the shortest path, as the measurement of path length.

3. **Call Handling**

   A host within the network is randomly chosen as sender. A number of calls are generated to simulate multicast request of this sender. For every call, the sender finds the shortest path to the core(s) of the multicast group. Then the multicast packet can be sent to all group members from the core via the shortest path tree. A call arrival will reserve its required bandwidth through the path to all group members. When a call departs, the reserved bandwidth will be freed. This is similar to the reservation process given by the RSVP protocol of the Internet ([11]). In every experiment, the same calls are created for CBT and MCBT to ensure the fairness of comparison.

4. **Computing Metrics**

   In every experiment, the values of the performance metrics for CBT and MCBT are computed multiple times with different seeds. Finally, we take the average of those values and compute 95% confidence intervals.

Two factors are varied in the simulation experiments: call bandwidth and group size. First we investigate the performance of MCBT on the same group structure under different traffic load. Step 2,3 and 4 are repeated with varied *call bandwidth*. Step 1 will not be repeated so different runs have the same group members, but different core(s) and different incoming traffic.

To investigate the scalability of MCBT and CBT, another experiment is designed where the group size is varied. It repeats Step 1,2,3 and 4. In Step 1, a different *size of group* is chosen in every run. Then in Step 3, the same calls are created to obtain the same incoming traffic for CBT and MCBT in

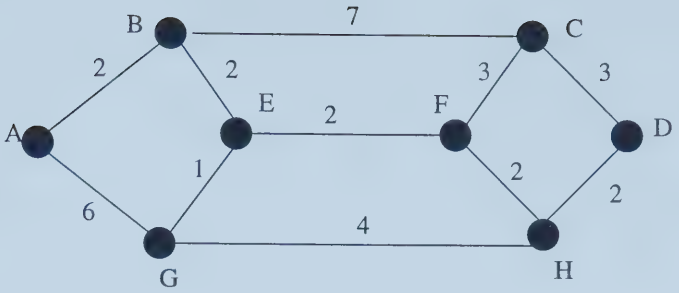


Figure 4.1: Simulation Verification

every run. Thus the performance of CBT and MCBT on the same incoming traffic but different group size is compared.

To investigate different scenarios of CBT and MCBT, in one experiment, we implement all scenarios. So in a single run, different scenarios will be investigated with the same incoming traffic and same multicast group. This will save the simulation time and make the comparison between different scenarios stand on the same ground.

4.6 Simulation Methodology

4.6.1 Simulation Verification

A small graph is generated to verify our simulator for CBT. This graph is shown in Figure 4.1. Suppose all hosts of this graph belong to the same group. Host A is chosen as sender while host B is a core. A queue of calls is generated as the incoming traffic to this group. The result from handwork matches the result from the simulator for the performance metric.

4.6.2 Transient State Removal

In this simulation, we are only interested in the *steady-state performance*, that is, the performance after the system has reached a stable state. For delay, blocking, bandwidth cost and core concentration, the initial part should be deleted, because the network is not fully loaded for traffic parameters until this point.

An experiment is conducted to investigate the performance of CBT as the

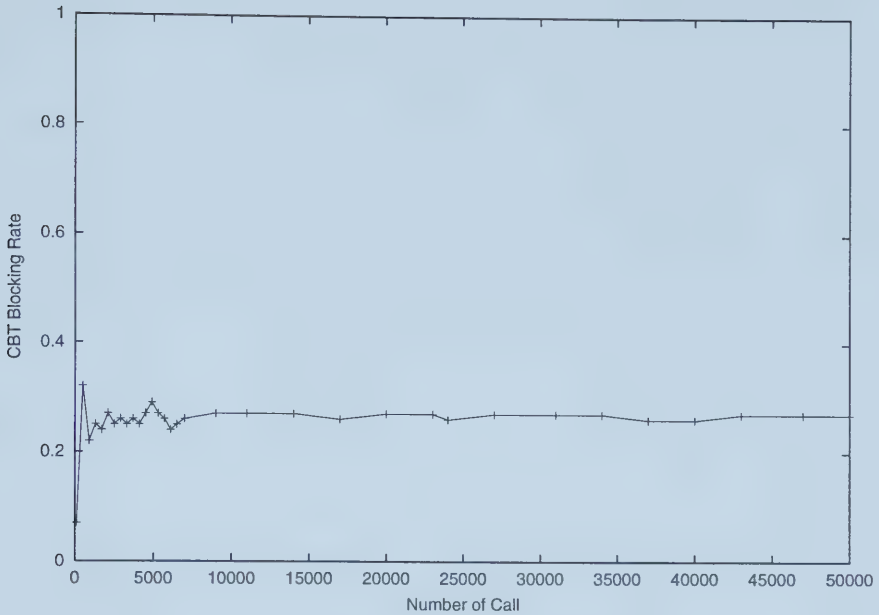


Figure 4.2: Transient Removal

length of simulation increases. In this experiment, we take a snap shot of blocking rate at different number of calls. By observing the blocking rate of CBT, we try to find the number of calls where the transient state ends (see Figure 4.2). From the plot we can see the transient state ends when the number of calls is $10,000$. In the following simulation, the initial data of the transient interval will be deleted to compute the performance metric we are interested in.

4.6.3 Stopping Criteria for the Simulation

To properly choose the length (number of calls) of simulation, another simulation is conducted. This simulation is similar to the one we designed to find transient state except the transient state in it has been removed.

Figure 4.3 shows that the CBT blocking rate is steady after $number of call = 100,000$. To be certain of the result, we run the simulations for $200,000$ calls.

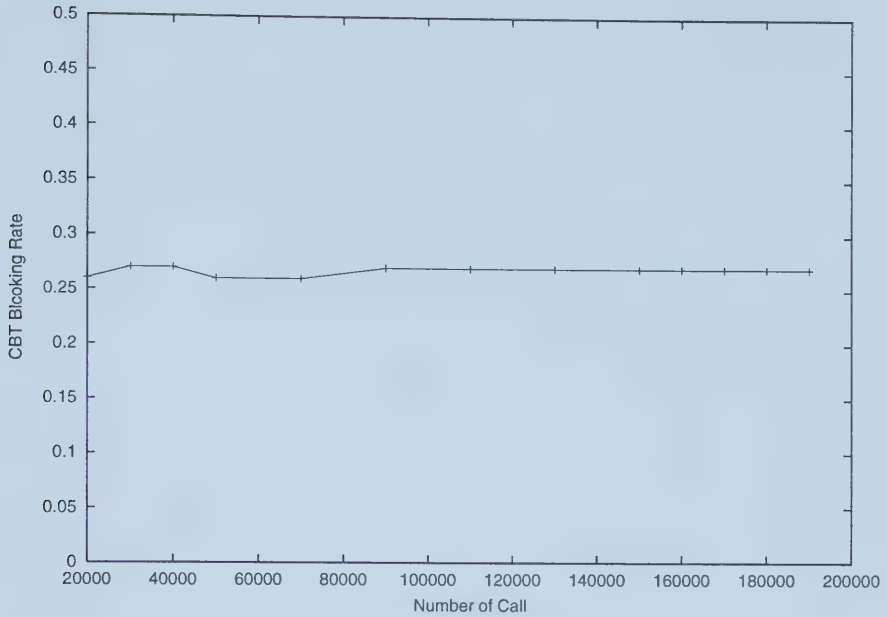


Figure 4.3: Simulation Termination

4.6.4 Data Analysis Method

As we have introduced in Section 4.5, we vary two factors: call bandwidth and group size, in the simulation. For each run, we replicate it 15 times with the same parameters setting but different random seed values. Then we average the results from different runs. The distribution of metric value is assumed as t distribution, and a two-sided 95% confidence interval is computed.

4.7 Quantitative Results

The evaluation approach of CBT and MCBT has been discussed in the previous section. Based on the two core selection scenarios that have been addressed in Section 4.1, a series of simulation experiments are designed to investigate the performance of CBT and MCBT. First we compared them when the cores are randomly chosen. Then we investigate their performance when the group topology information is available. Finally we compare the performance of MCBT(random) and MCBT(sparse).

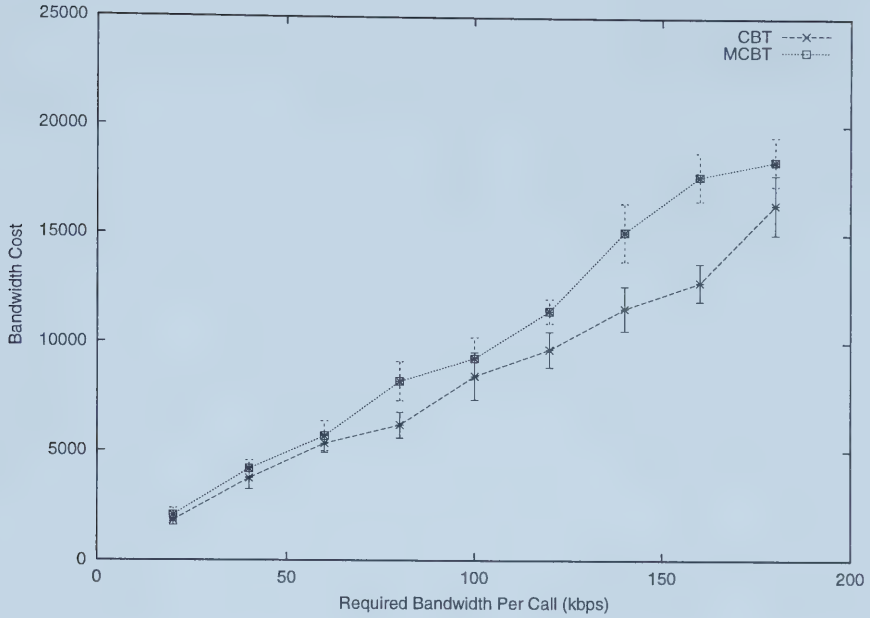


Figure 4.4: Bandwidth Cost on Various Call Bandwidth: CBT(random) vs. MCBT(random)

4.7.1 CBT vs. MCBT: Randomly Selected Core

In the first set of experiments, the *Randomly selected core* scenario is implemented to choose core(s) for CBT and MCBT. The motivation of this set of experiments is to investigate how robust CBT and MCBT are when group topology is not available.

An experiment is conducted to investigate the performance of CBT and MCBT with increasing traffic load by increasing the bandwidth requirement of incoming calls from 20kbps to 180kbps(see Section 4.5 for simulation process). The performance measures are shown in Figure 4.4 to Figure 4.7.

Figure 4.4 shows that the bandwidth cost of CBT and MCBT will increase linearly with the increase of required bandwidth. This is reasonable because all incoming calls go through the same shortest path tree from sender to all group members and the bandwidth cost is aggregated. The graph also shows that *MCBT has higher bandwidth cost than CBT.* This is because MCBT must maintain two shortest path trees, and therefore may not be able to merge bandwidth on links as much.

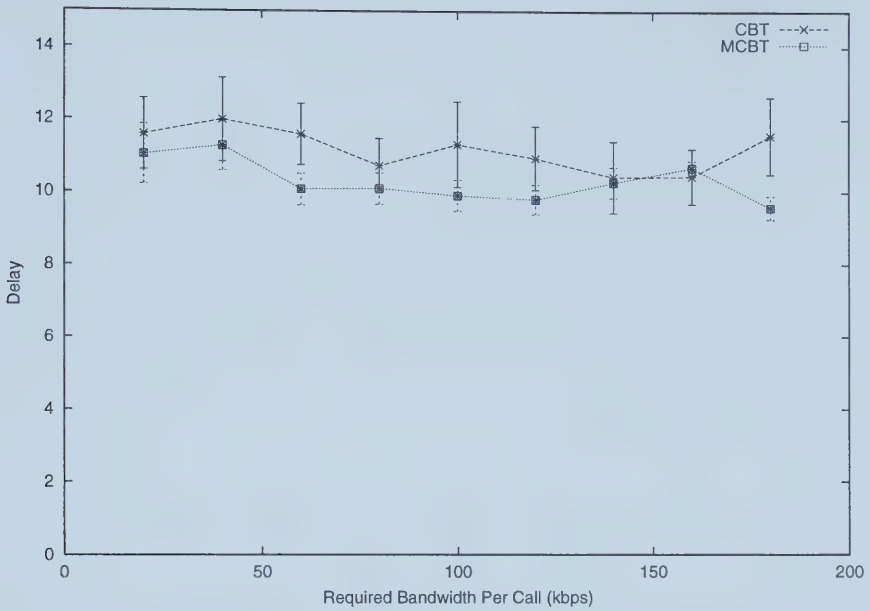


Figure 4.5: Delay on Various Call Bandwidth: CBT(random) vs. MCBT(random)

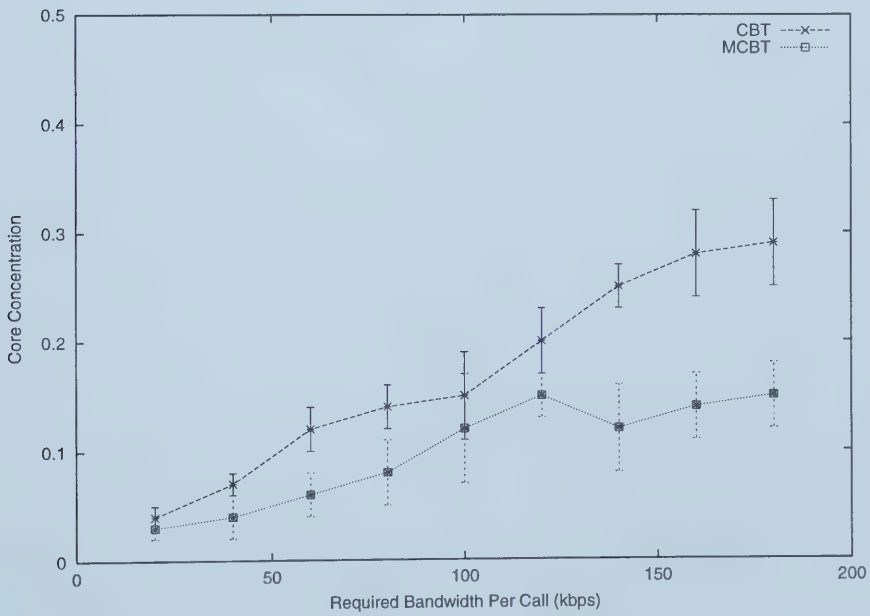


Figure 4.6: Core Concentration on Various Call Bandwidth: CBT(random) vs. MCBT(random)

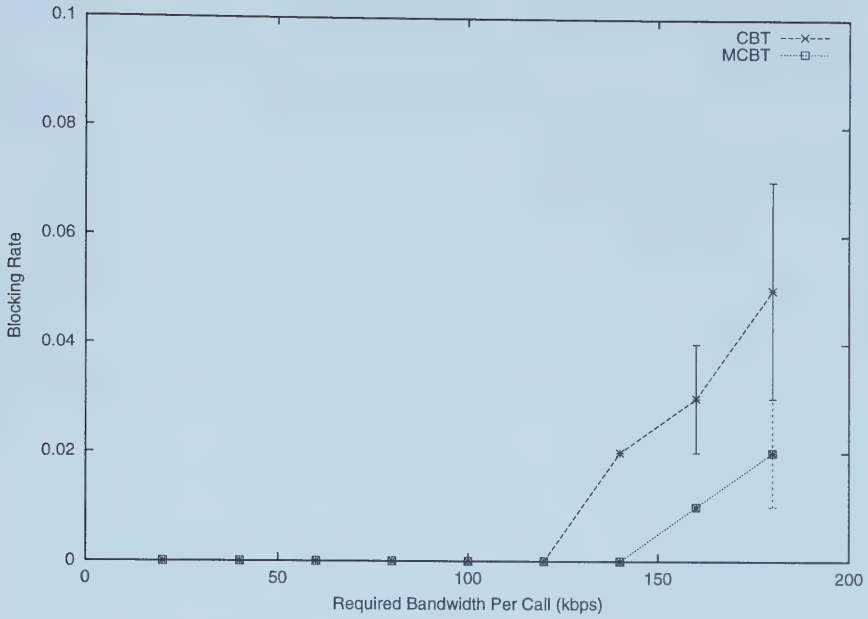


Figure 4.7: Blocking Rate on Various Call Bandwidth: CBT(random) vs. MCBT(random)

The average delay is shown in Figure 4.5. We can see that the delay of MCBT and CBT does not increase significantly with the increase of required bandwidth per call. The definition of delay in this simulation is the maximum geographical distance from sender to a most distance group member. So the delay greatly depends on the shape of the multicast tree but not on the incoming traffic load. Thus the selection of cores will mostly determine the delay. The experiment result shows that the delay of CBT and MCBT is highly variable. This is because the randomly chosen cores make the multicast tree change in every run. A randomly select core may be the *best core* or *worst core* in terms of delay, therefore it is important to see the effect of selecting in different cores. The simulation result shows: *in terms of delay, the performance of MCBT and CBT depends greatly on the core selection. It is difficult to determine which one would be better unless the core has been chosen. However, it does appear that MCBT has lower or similar delay to CBT.*

Figure 4.6 shows the core concentration of MCBT and CBT. For MCBT,

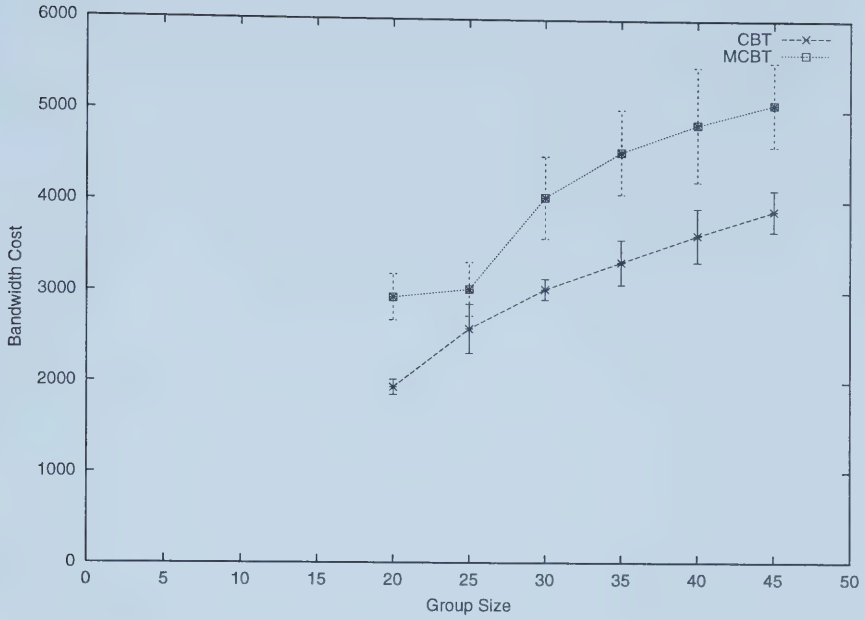


Figure 4.8: Bandwidth Cost on Various Group Size: CBT(random) vs. MCBT(random)

the core concentration is the average concentration of 2 cores. With the increase of incoming call bandwidth, the core concentration of CBT and MCBT increases. It is clear that: *compared to CBT, MCBT reduces the core concentration particularly for higher bandwidth requirement.* The reason is that the cores of MCBT handle fewer group members. This result meets the design purpose of MCBT.

Figure 4.7 shows the blocking rate of MCBT and CBT as bandwidth requirement increases. The result shows that *MCBT can greatly reduce the blocking rate.* This is because MCBT divides a multicast group into subgroups, and therefore reduces the core concentration. With fewer group members in a subgroup, the possibility of a bottleneck is lower. Thus the blocking rate is reduced.

After the investigation on the effect of different traffic load on MCBT and CBT, we investigate the scalability by varying the multicast group size(see Section 4.5 for simulation process). Figure 4.8 to Figure 4.11 show the performance of CBT and MCBT on group sizes from 20 to 45.

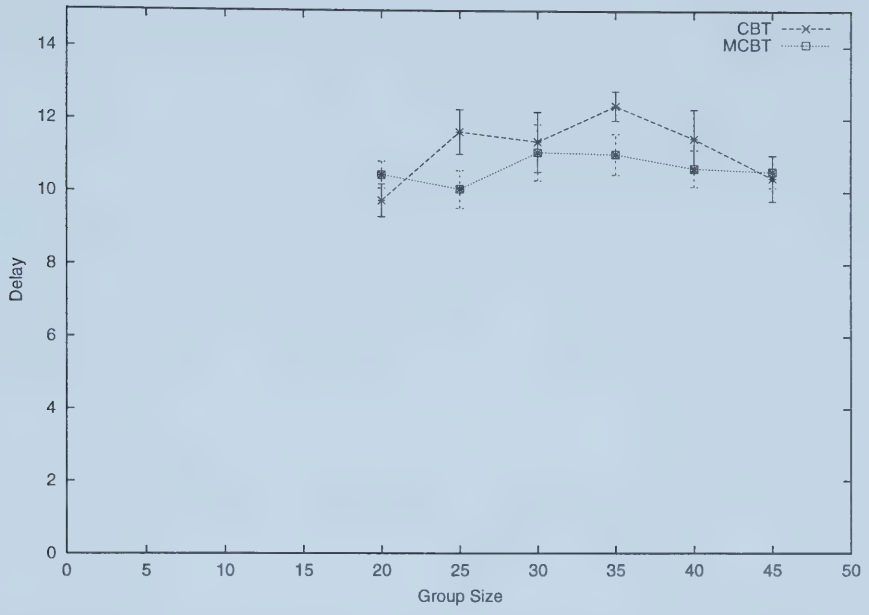


Figure 4.9: Delay on Various Group Size: CBT(random) vs. MCBT(random)

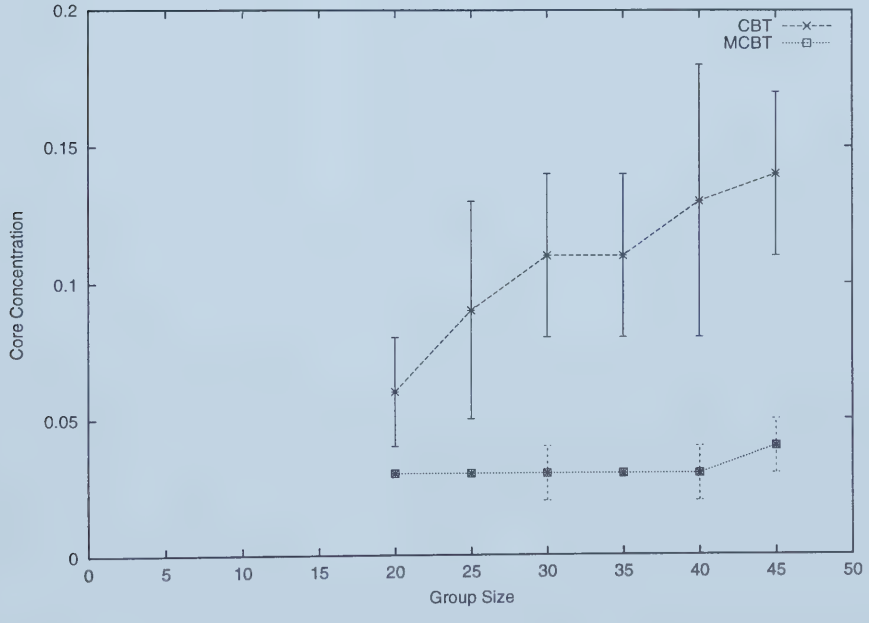


Figure 4.10: Core Concentration on Various Group Size: CBT(random) vs. MCBT(random)

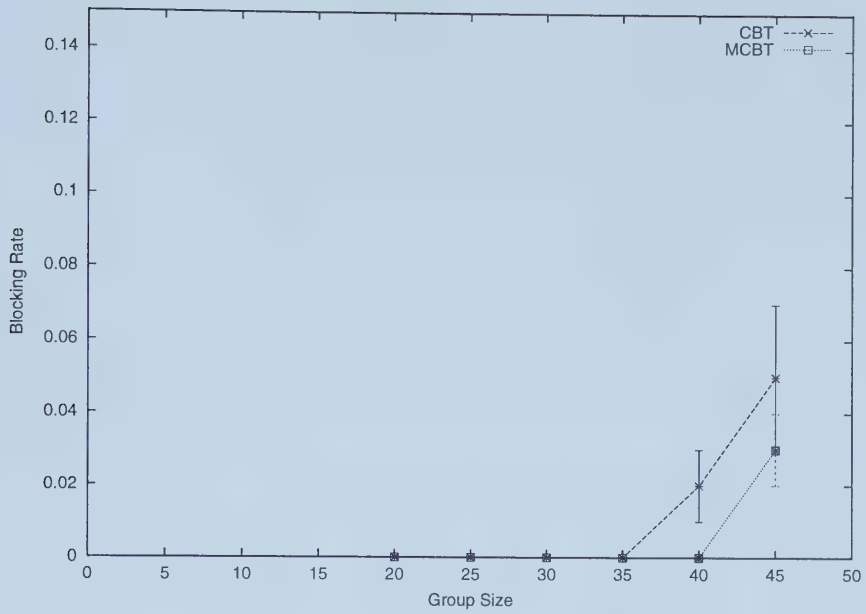


Figure 4.11: Blocking Rate on Various Group Size: CBT(random) vs. MCBT(random)

This simulation produces quite similar results to the previous one. We can see that MCBT consumes more bandwidth, but causes less core concentration, less blocking and possibly less delay. Figure 4.10 shows that the increase of group size leads to an increase of core concentration of CBT, but the core concentration of MCBT does not increase significantly. This shows that MCBT scales well for multicast group size. This is because MCBT divides the multicast group into two subgroups so that every core will handle fewer non multicast-capable hosts compared to the only core in CBT. This creates less duplication of incoming packets in every subgroup. With random scenario of CBT and MCBT, the average concentration of two cores should be leveraged by each other, thus makes the average core concentration of the two cores for MCBT is less variable than the single core concentration of a single selected core for CBT.

Experiment on Different Topologies

The previous experiment was done on one random graph model with 100 nodes. In case this biased the results, we created two other graphs with the same network paramenters(100 nodes, etc. See Section 4.4.1) as the graph we just used, and performed this experiment on these graphs. We chose the same simulation parameters as in Section 4.7.1 to create the same incoming traffic. The simulation results are shown in Table 4.3 and Table 4.4. These two tables demonstrate that under different graph topology, the relative performance of MCBT and CBT is similar.

	MCBT(random)	CBT(random)
graph1	1856	1512
graph2	1273	1028
graph3	2032	1633

Table 4.3: Comparison of bandwidth cost on different networks

	MCBT(random)	CBT(random)
graph1	11.04	11.59
graph2	13.32	12.52
graph3	12.14	13.29

Table 4.4: Comparison of delay on different networks

Summary

This experiment investigates the randomly selected core scenario. From the results, MCBT shows better scalability than CBT. While it may have relatively higher bandwidth cost, it can reduce core concentration and therefore reduce the bottleneck of CBT. Thus we draw the following Summary: *a randomly chosen core among group members with no knowledge of the group structure can provide desirable performance in terms of scalability for MCBT over CBT.*

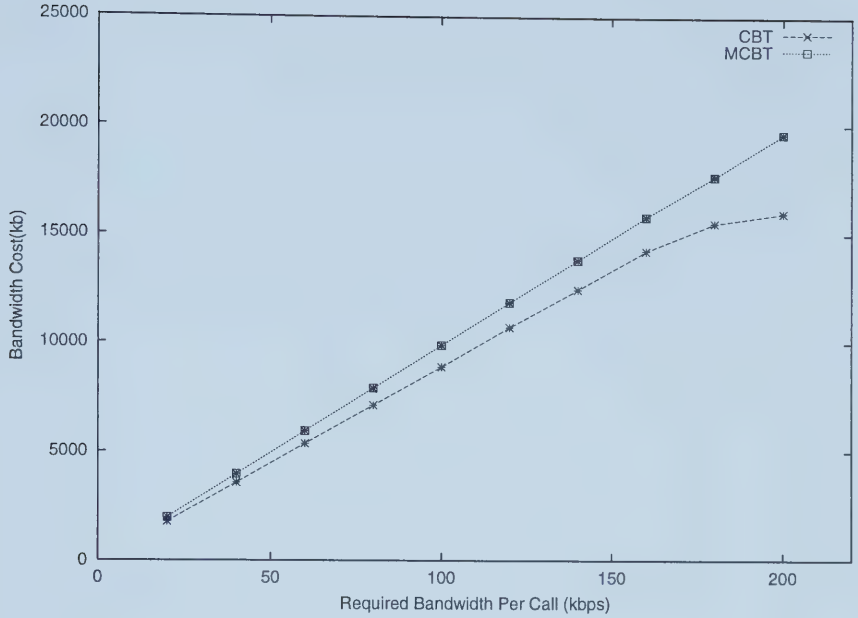


Figure 4.12: Bandwidth Cost on Various Call Bandwidth: CBT(DCT) vs. MCBT(sparse)

4.7.2 CBT(DCT) vs. MCBT(sparse)

In the next set of experiments, the performance of MCBT(sparse) is investigated and compared to CBT(DCT). We want to see how MCBT and CBT perform when the group topology information is used. In this set of experiments, once a group is created, because of the core selection scenario, the cores will be the same for each run (see Section 4.1). The simulation result shows that once the core is fixed, the confidence intervals are narrow.

In the first experiment, the bandwidth requirement for each call increases from 20kps to 180kps(see Section 4.5 for simulation process). Figure 4.12 to Figure 4.15 show the performance of CBT(DCT) and MCBT(sparse).

Figure 4.12 shows that the bandwidth cost of CBT(DCT) and MCBT(sparse) will increase linearly with the increase of require bandwidth per call; while Figure 4.13 shows that delay of CBT(DCT) and MCBT(sparse) keeps steady. This result is similar to that of CBT(random) and MCBT(random). This is because in every run, the multicast tree will be the same, then the hop count to send a multicast packet from core to all group members will be same for

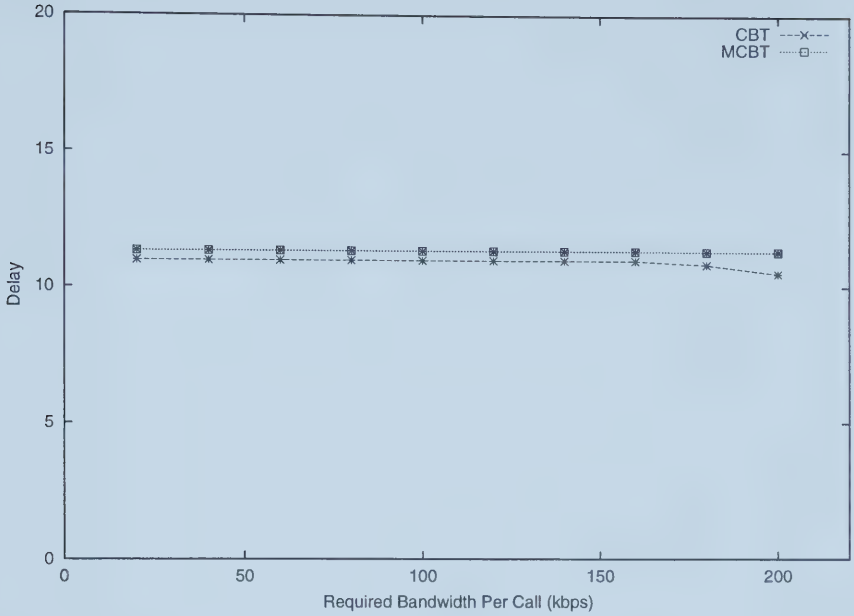


Figure 4.13: Delay on Various Call Bandwidth: CBT(DCT) vs. MCBT(sparse)

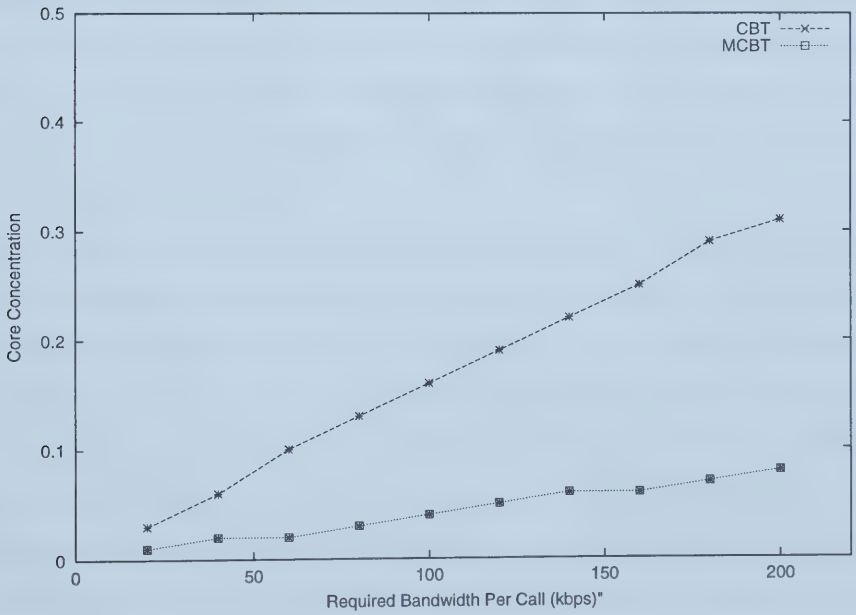


Figure 4.14: Core Concentration on Various Call Bandwidth: CBT(DCT) vs. MCBT(sparse)

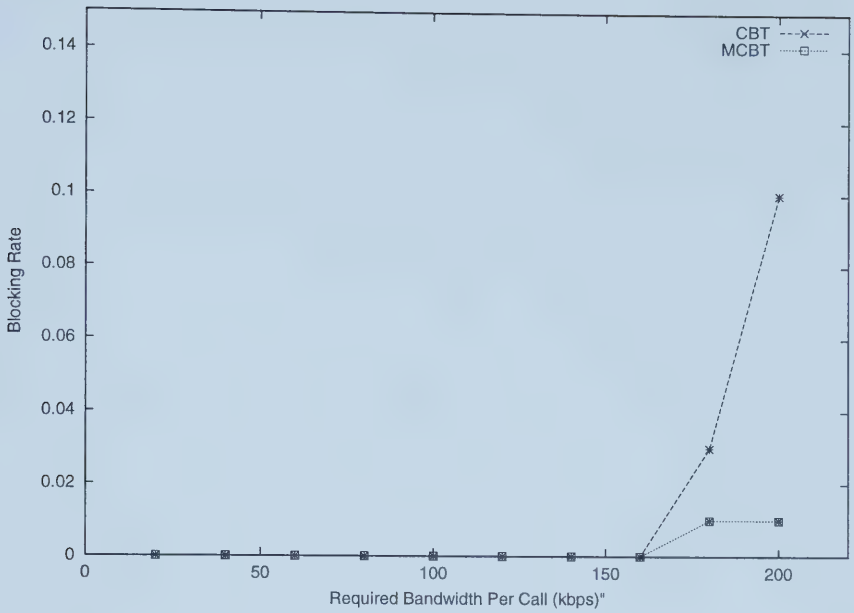


Figure 4.15: Blocking Rate on Various Call Bandwidth: CBT(DCT) vs. MCBT(sparse)

the same algorithm. Thus the total bandwidth cost will increase linearly with increased required bandwidth. Since delay depends only on the construction of multicast tree, it will not increase with the increase of required bandwidth. Figure 4.12 also demonstrates that MCBT(sparse) costs relatively more bandwidth (about 10%) than CBT(DCT), this is because MCBT(sparse) has two multicast trees to maintain.

Figure 4.14 shows that the relative performance of MCBT(sparse)and CBT(DCT) on core concentration is similar to that of MCBT(random) and CBT(random). We can see that MCBT(sparse) has significantly lower core concentration than CBT(DCT), especially when the required bandwidth is high. This is because with two cores handling the multicast group, every core will have smaller multicast trees to deliver to. This clearly demonstrates that MCBT has better performance in terms of core concentration when the traffic load is increasing.

Figure 4.15 shows that MCBT(sparse) has a lower blocking rate than CBT(DCT). This is because MCBT(sparse) divides the multicast group into two subgroups, making the traffic load on each subgroup less than that of

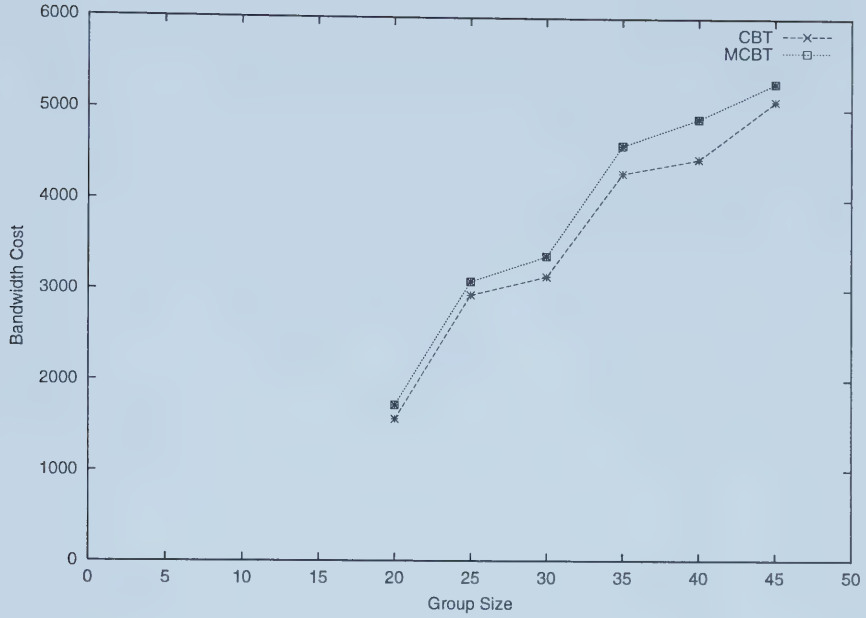


Figure 4.16: Bandwidth Cost on Various Group Size: CBT(DCT) vs. MCBT(sparse)

CBT(DCT). Therefore the possibility of bottleneck in MCBT(sparse) is less than that of CBT(DCT).

The simulation results indicate that MCBT(sparse) is scalable. It can maintain good performance on delay, core concentration and blocking rate under increasing required bandwidth. Although MCBT(sparse) may cost slightly more bandwidth than CBT(DCT), it can greatly reduce core concentration and blocking rate.

Another experiment is designed to investigate MCBT(sparse) performance with different group size(see Section 4.5 for the simulation process). Figure 4.16 to Figure 4.19 show the simulation results for different metrics on a increasing group size from 20 to 45. In this experiment, we use the same parameter setting as in the experiment on CBT(random) and MCBT(random)(Section 4.7.1). We can see that these two experiments produce relatively similar results, but the confidence interval tends to be very narrow in MCBT(sparse) and CBT(DCT).

Figure 4.16 shows that the bandwidth cost will increase when group size

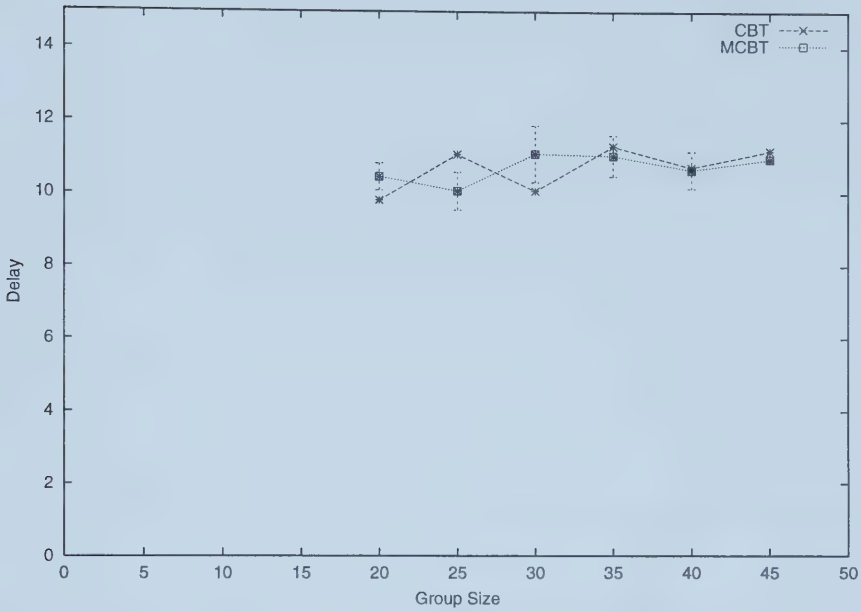


Figure 4.17: Delay on Various Group Size: CBT(DCT) vs. MCBT(sparse)

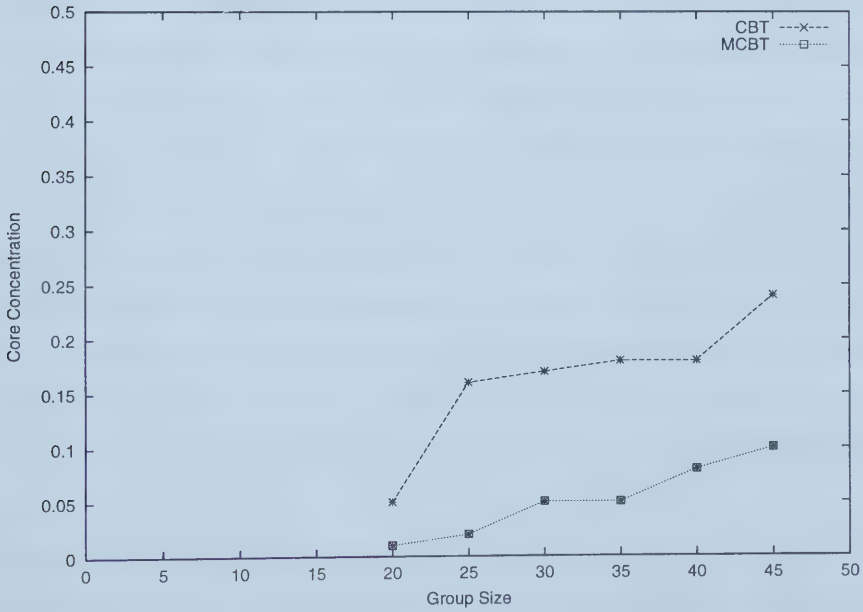


Figure 4.18: Core Concentration on Various Group Size: CBT(DCT) vs. MCBT(sparse)

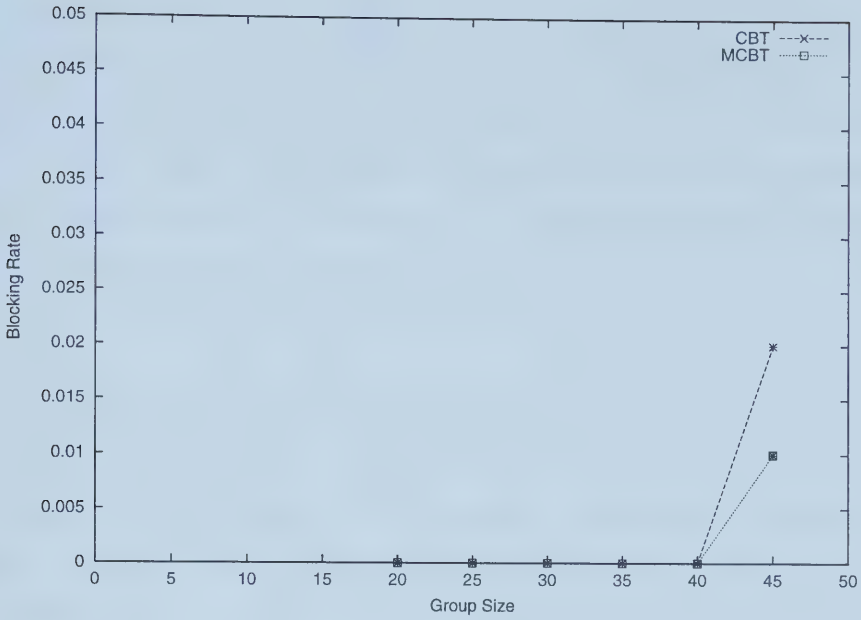


Figure 4.19: Blocking Rate on Various Group Size: CBT(DCT) vs. MCBT(sparse)

increases. This is reasonable because more group members lead to more bandwidth usage. We can also see that MCBT(sparse) has relatively higher bandwidth cost(about 7%), this is because MCBT has two multicast trees to maintain, and therefore may not be able to merge bandwidth cost on links as much. Figure 4.17 shows that the delay for CBT(DCT) and MCBT(sparse) keeps steady with an increasing group size. This indicates that the maximum geographical distance from core to multicast group members will not change significantly upon the changing group size. Figure 4.17 also demonstrates that MCBT(sparse) and CBT(DCT) produce relatively similar delay. The performance on delay demonstrates that CBT(DCT) and MCBT(sparse) have good scalability; while MCBT(sparse) may have slightly higher bandwidth cost than CBT.

Figure 4.18 and Figure 4.19 show that MCBT(sparse) has better performance than CBT(DCT) in terms of core concentration and blocking rate. As mentioned before, this is because MCBT(sparse) have two cores and subgroups. Every core and subgroup handles less traffic than CBT(DCT), thus

the core concentration and blocking rate will be reduced.

This experiment shows that *MCBT(sparse)and CBT(DCT) both can maintain a good performance on an increasing multicast group size. MCBT(sparse) and CBT(DCT) have similar delay that scales well with increasing group size. MCBT(sparse) has better performance on core concentration and blocking, while costing slightly more bandwidth.*

Summary

This set of experiments investigates the performance of MCBT(sparse) and CBT(DCT). The results show that MCBT(sparse) and CBT(DCT) both can maintain a good performance with increasing incoming traffic load and multicast group size. The results are less variable than the random case. The comparison between MCBT(sparse) and CBT(DCT) indicates: *if the group topology is available, MCBT(sparse) is a better choice than CBT(DCT) especially with high incoming traffic and large multicast group size. MCBT(sparse) can greatly reduce the core concentration and blocking rate of CBT(DCT), while it may cost slightly more bandwidth.*

4.8 MCBT: Further Discussion

In the previous two sections, we have already investigated the performance of CBT versus MCBT. Thus after the comparison between MCBT and CBT, now we pay more attention to MCBT. First we consider the core fairness between the two cores of MCBT in different scenarios, then we compare the performance of MCBT(sparse) and MCBT(random).

4.8.1 MCBT: Core Fairness

In MCBT, two multicast group members are chosen as cores (core1 and core2). It would be reasonable to consider the fairness of load of these cores. We want to have a balanced traffic load for each core. The traffic load of each core can be evaluated by core concentration. An experiment is conducted to see the core concentration of core1 and core2. In this experiment, first we create a

multicast group and then build multicast trees for two subgroups. Then 10 runs with the same call parameters but different random seeds are created. We compute the core concentration of each core on these 10 traffic. We want to see how is the performance of these two cores when the multicast group, subgroup, incoming call parameter are the same, while we use different random seed. Table 4.5 and Table 4.6 show the simulation result. (In these two tables, we always show the core with greater core concentration as core1)

run	1	2	3	4	5	6	7	8	9	10
core1	0.27	0.14	0.20	0.01	0.24	0.19	0.06	0.03	0.01	0.09
core2	0.05	0.05	0.06	0.00	0.00	0.02	0.01	0.02	0.00	0.08

Table 4.5: Comparison of Core Concentration for MCBT(random) 1

run	1	2	3	4	5	6	7	8	9	10
core1	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14
core2	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12

Table 4.6: Comparison of Core Concentration for MCBT(sparse) 1

To be certain that the results are not affected by the multicast group structure, we repeat this experiment on another multicast group with the same group size. The results are shown in Table 4.7 and Table 4.8.

run	1	2	3	4	5	6	7	8	9	10
core1	0.06	0.14	0.18	0.12	0.24	0.10	0.21	0.23	0.11	0.01
core2	0.04	0.06	0.08	0.04	0.07	0.02	0.08	0.02	0.04	0.00

Table 4.7: Comparison of Core Concentration for MCBT(random) 2

Table 4.5 and Table 4.7 clearly demonstrate that random choice of cores does not produce balanced load for the 2 cores. For MCBT(random), the cores are randomly chosen, thus the cores will not be the same pair every time. All multicast group members choose the nearest core to join. So the size of two subgroups may vary a lot. Thus the traffic load of the two cores may be highly unbalanced.

run	1	2	3	4	5	6	7	8	9	10
core1	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10
core2	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08

Table 4.8: Comparison of Core Concentration for MCBT(sparse) 2

Table 4.6 and Table 4.8 show that MCBT(sparse) can give a balanced traffic load to these two cores. MCBT(sparse) always pick a pair of hosts who have maximum distance. This ensures the size of two subgroups is more similar. Thus the load of two cores is more balanced.

Since MCBT(sparse) is running on the same multicast group for a particular experiment, the cores of MCBT(sparse) will be the same on different runs. To be certain MCBT(sparse) can keep its core fairness on different multicast group structures, another experiment is designed. In every run of this experiment, we keep the incoming traffic and the multicast group size the same, but the multicast group members randomly change. This ensures that the cores of each run are different. The result of this experiment is shown in Table 4.9. It shows that MCBT(sparse) can always keep the load of two cores relatively balanced on different group topology.

run	1	2	3	4	5	6	7	8	9	10
core1	0.13	0.10	0.12	0.14	0.15	0.18	0.14	0.10	0.14	0.12
core2	0.10	0.09	0.09	0.12	0.12	0.15	0.12	0.09	0.11	0.10

Table 4.9: Comparison of Core Concentration for MCBT(sparse) 3

Summary

The experiments show that MCBT(sparse) can give more balanced traffic load to the two cores than MCBT(random).

4.8.2 MCBT(random) vs. MCBT(sparse)

As we have introduced in Section 4.5, our experiments simulate four scenarios of MCBT and CBT at the same time. In Section 4.7.1 we investigate random scenarios of MCBT and CBT. In Section 4.7.2, we compare MCBT(sparse)

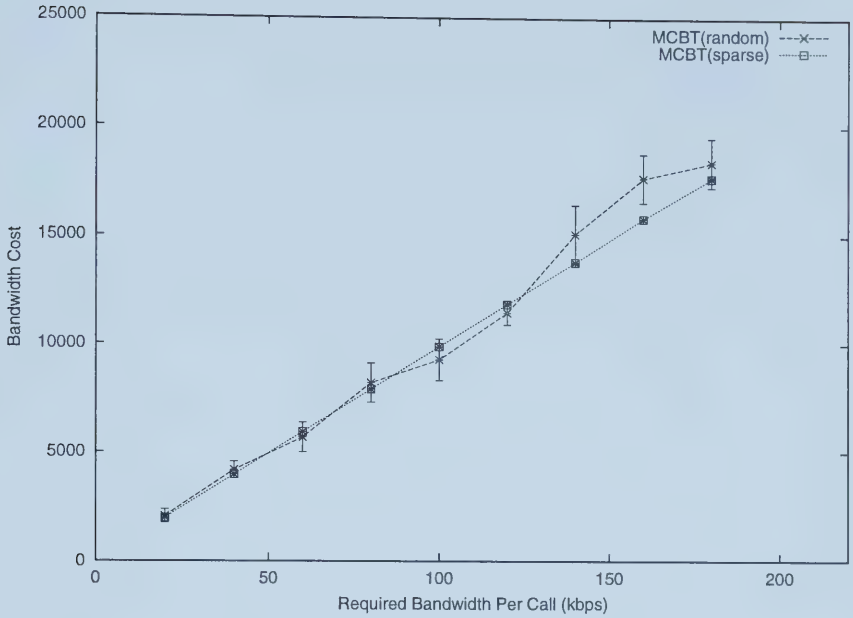


Figure 4.20: Bandwidth Cost on Various Call Bandwidth: MCBT(random) vs. MCBT(sparse)

and CBT(DCT). With the high variation of random scenarios, graphs are hard to read if we put these four scenarios together. Thus we compare two related scenarios at one time. In this section, we compare MCBT(random) and MCBT(sparse) to see if it is useful to implement a mechanism that is more complex than random choice.

Figure 4.20 to Figure 4.23 compare the performance of MCBT(sparse) and MCBT(random) for different call bandwidth. Figure 4.24 to Figure 4.27 compare them on various group size.

MCBT(sparse) is more complex than MCBT(random). It needs all group topology information to find the *best* cores for all subgroups; while MCBT(random) just randomly chooses cores among the subgroup members. Thus MCBT(sparse) has more set-up cost. Once the multicast trees are built, operationally the cost to maintain the multicast tree of MCBT(sparse) is not more than that of MCBT(random). Every sender just needs to find the cores and every core needs to manage its subgroup members.

The comparison shows that MCBT(sparse) has less variable performance

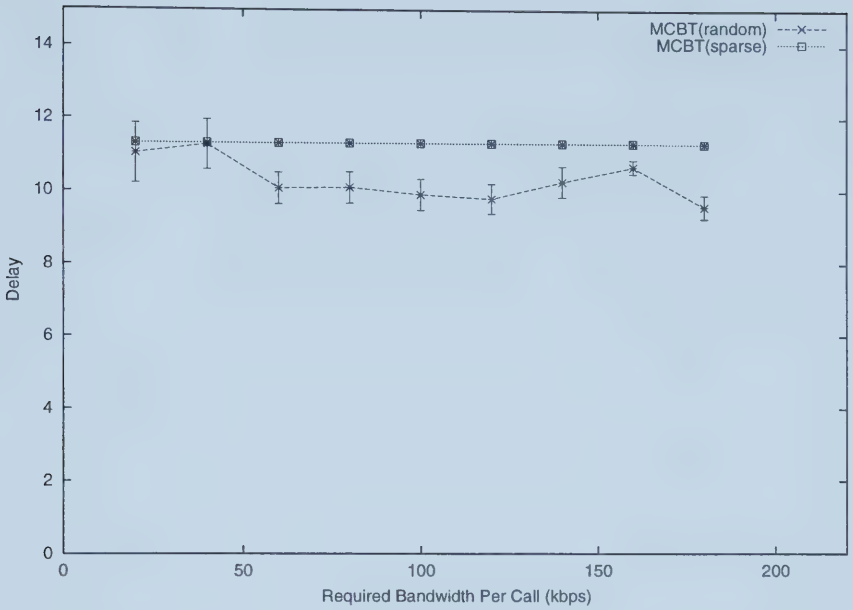


Figure 4.21: Delay on Various Call Bandwidth: MCBT(random) vs. MCBT(sparse)

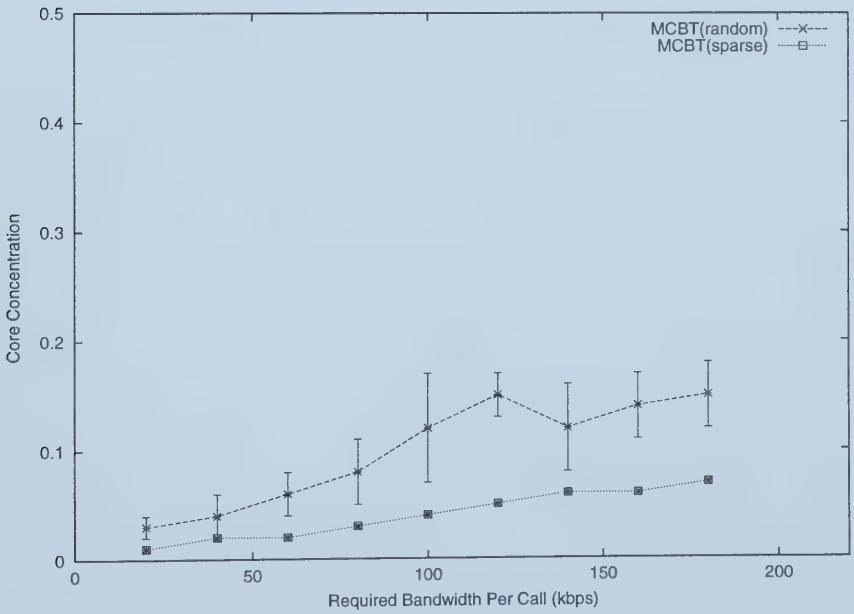


Figure 4.22: Core Concentration on Various Call Bandwidth: MCBT(random) vs. MCBT(sparse)

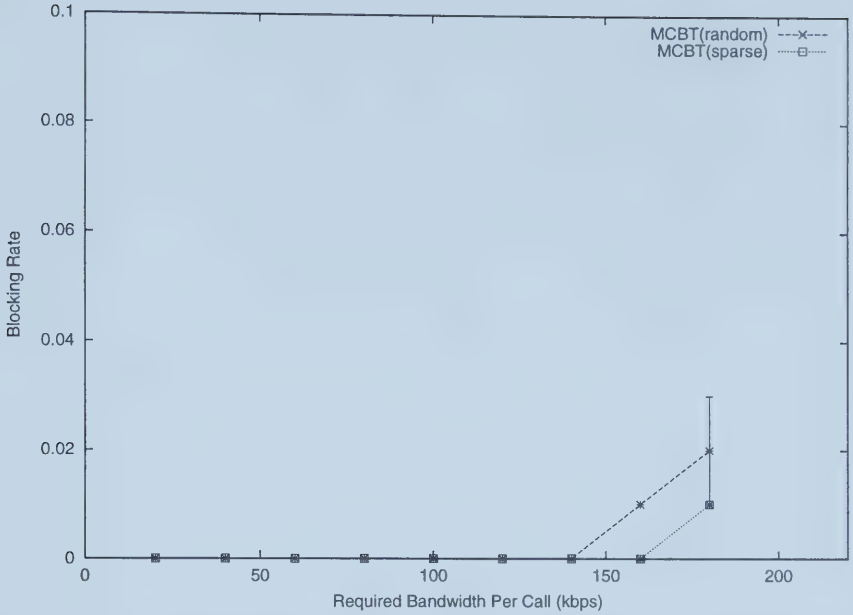


Figure 4.23: Blocking Rate on Various Call Bandwidth: MCBT(random) vs. MCBT(sparse)

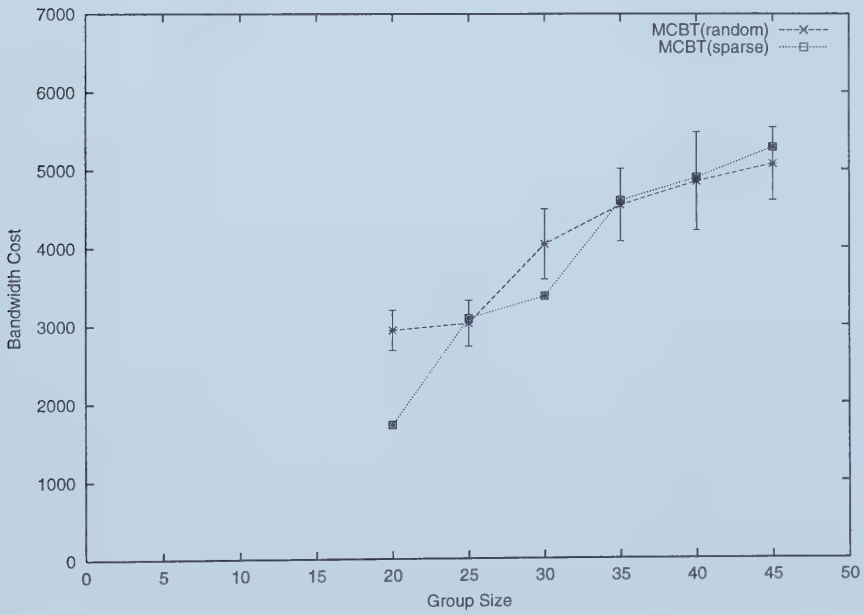


Figure 4.24: Bandwidth Cost on Various Group Size: MCBT(random) vs. MCBT(sparse)

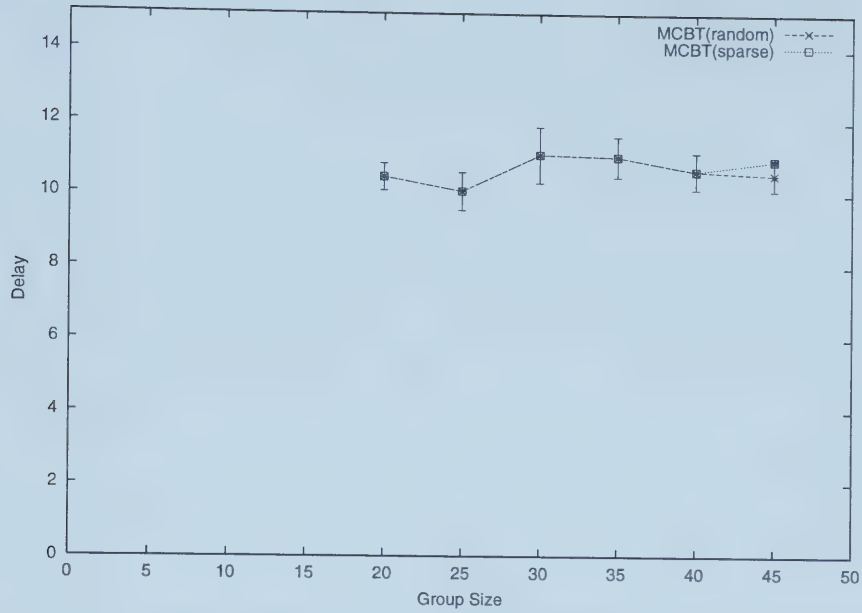


Figure 4.25: Delay on Various Group Size: MCBT(random) vs. MCBT(sparse)

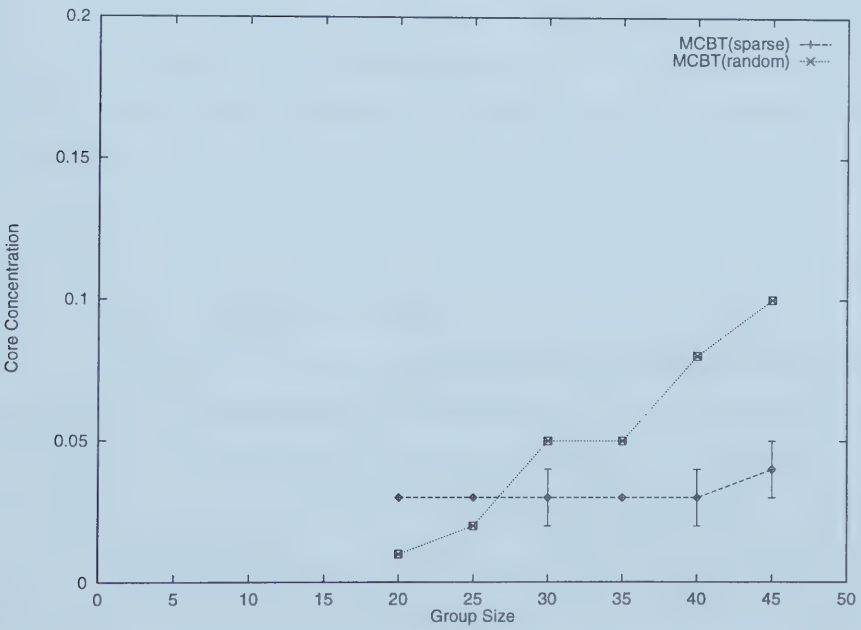


Figure 4.26: Core Concentration on Various Group Size: MCBT(random) vs. MCBT(sparse)

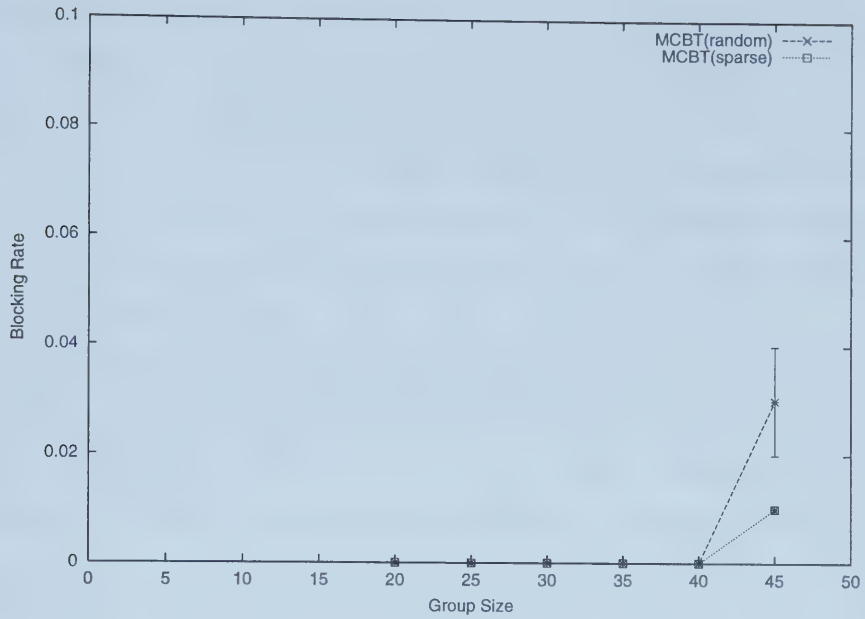


Figure 4.27: Blocking Rate on Various Group Size: MCBT(random) vs. MCBT(sparse)

than MCBT(random). This is because MCBT(random) randomly selects two group member as cores for each run, while MCBT(sparse) has one choice.

The average bandwidth of MCBT(sparse) cost is very close to MCBT(random) except for small group size.(see Figure 4.20 and Figure 4.24). In terms of delay, it is greatly depends on the shape of multicast tree. The delay is very similar for both choices (see Figure 4.21 and Figure 4.25). However, when group size is fixed, random choice may produce slightly less (average 7%) delay, which deserves more investigation in the future.

When the group size is large and incoming traffic increases, MCBT(sparse) produces lower core concentration than MCBT(random). This is shown in Figure 4.22 and Figure 4.26. This is because MCBT(sparse) allocates the cores sparsely. When the group size is large, we can imagine a case where both randomly selected cores happen to be located at one end of the graph and all group members are located throughout the graph. Then two sparsely located cores are more likely get two localized multicast trees; while randomly selected cores may get two sparse trees to maintain. Then the core concen-

tration will be higher for MCBT(random). When the group size is small, this may be different. The results show that core concentration can be lower for MCBT(random). MCBT(sparse) may choose two group members which are far away from most of the other group members; while the randomly scenario is more likely to choose two group members which are closer to other group members. In this case, MCBT(sparse) will have more core concentration. This can be seen in Figure 4.26.

Figure 4.23 and Figure 4.27 shows MCBT(sparse) can reduce blocking rate. The reason for this is that sparsely located cores can balanced the load of the two subgroups. With the same incoming traffic, a balanced load in two subgroups can reduce bottlenecks (see experiment in Section 4.8.1).

Summary

The comparison of the performance of MCBT(sparse) and MCBT (random) indicates: *MCBT(sparse) and MCBT(random) both scale well on increasing incoming traffic load and multicast group size. With the information of group topology, although MCBT(sparse) has more multicast tree set-up cost, it can provide better core concentration and blocking rate when the group size is large and incoming traffic is high. Also the bandwidth cost and delay of MCBT(sparse) and MCBT(random) are close. Finally, the results of MCBT(sparse) are less variable. These make MCBT(sparse) a good choice when we select multicast routing protocols.*

4.9 Conclusion

In this chapter, a series of simulation experiments were conducted to evaluate MCBT. The analysis of simulation result shows that MCBT meets our expectation. We can see that MCBT achieves good performance under increasing traffic load and multicast group size.

Our experiments show:

- MCBT and CBT both scale well on increasing incoming traffic load and multicast group size. The delay and blocking rate will not increase a lot

when the traffic load increases or multicast group size increases.

- Compared to CBT, MCBT has similar delay and relatively more bandwidth cost , but it can greatly reduce the traffic concentration and blocking rate.

- The experiments on two scenarios of MCBT(random and sparse)indicate that randomly selected cores can achieve good performance but random choice produces more variable results. If the group topology information is available, two sparsely located cores can improve the performance with balanced core concentration and less blocking when the group size is large and incoming traffic is high.

Chapter 5

Conclusion

This thesis gives an overview of relevant algorithms and protocols for IP multicast. After an evaluation and classification of these multicast protocols, a new approach for multicast over large area network, multiple core based tree (MCBT), is proposed. The motivation of MCBT is to improve the CBT protocol by reducing traffic concentration; while keeps CBT's high scalability. A series of experiments are designed to investigate the performance of MCBT. The experiment results show MCBT meets our expectation.

5.1 Contribution

We have investigated relevant algorithms and protocols for IP multicast. According to the type of multicast tree they build, we can classify currently used protocols for multicast into two categories: *source-based tree* and *group-shared tree*. The source-based protocols lack scalability, while group-shared protocols may cause high traffic concentration.

This thesis suggests MCBT to reduce the traffic concentration of CBT without losing its scalability. The operation and implementation of MCBT are addressed. From the simulation study and performance analysis we can see:

1. MCBT has high scalability. Our simulation investigates the performance of MCBT when the group membership is static. The experiments result shows that MCBT can scale well when the incoming traffic and multicast group size are increasing. If the group membership is dynamic, the design

of MCBT still ensure its high scalability just as CBT. If a host wants to join a multicast group, it only needs to send request to the root. The root will reply it with the nearest core. Then the sender sends a join request message to that core, and the core adds a new branch to the multicast tree it maintains. If it a host wants to leave a group, after receiving the leave request message from the host, the core prunes the branch to that host. So the overhead of joining/leaving is low for MCBT.

2. The simulation demonstrates that MCBT performs well with increasing traffic load and group size, which meet our expectation. The increasing rate of the delay, core concentration and blocking rate is far less than that of incoming traffic load or multicast group sizes. Compared to CBT, MCBT has similar delay and slightly higher bandwidth cost, but can greatly reduce traffic concentration and blocking. Furthermore, we can reduce the blocking rate by sparsely choosing cores.

3. Two scenarios are suggested to implement MCBT. We may randomly choose two group members as cores, or choose two group members who are located as far as possible. These two scenarios are called MCBT(random) and MCBT(sparse). Compared to the sparse scenario, the random scenario does not require the multicast group topology information and the set up cost to find cores is low. Random scenario can give good performance if the group topology information is not available. With information of group topology, sparsely chosen cores can improve the performance of MCBT with balanced core concentration and less blocking.

5.2 Future Work

However, MCBT is still in its early definition. It establishes a basis for more research and implementation work.

1. The bandwidth cost of MCBT is higher than that of CBT. This may harm the total performance of MCBT if the bandwidth resource is scarce.

Further study is demanded for this limitation.

2. How many cores should be chosen for a particular group? In our research, we choose two cores. We can imagine that the more cores we choose, the less core traffic concentration, the more group management cost and the more complex the sender connection will be. We need to find a method to choose the core number, reduce group traffic concentration while minimizing group management cost. This method should be easy to implement. As a preliminary idea, we can consider the relation between group size and core number.

3. How can we build multiple core based trees from scratch? If we have already randomly select cores, with the group size increasing, these cores may keep their roles. But if the cores are chosen by some other criteria, then the cores may not meet this criteria when the group members join/leave. The old cores may not be optimal, but the cost to reselect cores and rebuild multicast trees will be high. When and how should we this reconstruction?

4. Other core choice method. In this simulation, we suggest a core choice method to balance the load of subgroups. However, other method need to be found if we want a lower bandwidth cost or delay.

5. The biggest challenge is to implement MCBT in real network, which is also the most interesting part of research. First we want to use it on Internet according to the properties and needs of Internet. After successful test on Internet, we can transfer to another platform: ATM. We can see how well MCBT does for IP multicast over ATM.

Bibliography

[1] S. Deering, "Host extensions for IP multicasting ",STD 3, RFC1112, Stanford University, August 1989 http://www.internic.net/rfc/rfc1112.txt.

[2] "IP Multicast White Paper Serier" http://www.ipmulticast.com/community/whitepapers.

[3] W. Fenner, "Internet Group Management Protocol, Version 2", Xerox Parc January 1997.

[4] B. Cain, A. Thyagarajan, and S. Deering, "Internet group management protocol, version 3", Internet Draft draft-cain-igmp-00.txt, Expires March 8, 1996.

[5] Y. Dalal and R. Metcalfe "Reverse path forwarding of broadcast packets", Communications of the ACM, Vol. 21, December 1978, pp. 1040-1048.

[6] E. W. Dijkstra "A note on two problem in connection with graphs", Numerical Mathematics, October 1959.

[7] L. Wei and D. Estrin, "A comparison of multicast trees and algorithms", TR USC-CD-93-560, Dept. Computer Science, University of California, Sept 1993 for a comparison of heuristic approaches.

[8] R. M. Karp, "Reducibility among combinatorial problem". Plennum Press, New York, 1972.

[9] D. Waitzman, C. Patridge, and S. Deering. "Distance vector multicast routing protocol", RFC-1075, November 1988.

[10] A. Ballardie, P. Francis, and J. Crowcroft. "Core based trees (CBT): an architecture for scalable inter-domain multicast routing", Proceedings of ACM SIGCOMM'93, San Fransico, August 1993.

[11] R. Braden, Ed., "Resource ReServation Protocol (RSVP)- version 1 Functional Specification", RFC 2205, September 1997.

[12] K. Calvert, E.Zegura, and M.Donahoo, "Core selection methods for multicast routing", ICCCN'95.

[13] J.Moy "Multicast extensions to OSPF", RFC-1584, March 1994.

[14] J.Moy "OSPF version 2", RFC-1583, March 1994.

[15] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C.-G. Lue, and L. Wei. "An architecture for wide-area multicast routing", Proceedings of ACM SIGCOMM'94, London, September 1994.

[16] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: a transport protocol for real-time applications", RFC 1889, January 1996, http://www.internic.net/rfc/rfc1889.txt.

[17] A. Thyagarajan and S. Deering, "Hierarchical distance-vector multicast routing for the MBone", in Proceeding of ACM SIGCOMM'95, 1995.

[18] ATM Forum, "ATM user network interface (UNI) specification version 3.1", ISBN 0-13-393828-X, Prentice Hall, Englewood Cliffs, NJ, June 1995.

[19] ATM Forum, "ATM user network interface (UNI) signaling specification version 4.0", ATM Forum: af-sig-0061.000, July 1996.

[20] Norihiro Ishikawa, "IP multicast routing over ATM", IETF Internet Draft: draft-ishikawa-ion-mcatm-routing-00.txt, July 1997.

[21] Armitage, G.J., "Support for multicast over UNI3.0/3.1 based ATM networks", Internet Draft, draft-ietf-ipatm-ipmc-12.txt, Feb 1996.

[22] Talpade, R.R., and Ammar, M.H., "Multicast server architectures for MARS-based ATM Networks", Internet Draft, draft-talpade-ion-marsmcs-01.txt, July 1996.

[23] E. Zegura, Kenneth L. Calvert, Samrat Bhattacharjee, "How to model an Internetwork", Proceedings of IEEE Infocom '96, San Francisco, CA.

[24] Thaler, D.G. and Ravishankar C.V. "Distributed center-location algorithms: Proposals and Comparisons." In IEEE Infocom '96. IEEE, March 1996.

[25] M.Doar and I. Leslie. "How bad is naive multicast routing", In Proceeding of the IEEE Infocom'93,1993.

[26] Talpade, and Ammar, "Multiple MCS support using an enhanced version of the MARS server", Internet Draft, draft-talpade-ion-multiplemcs-01.txt, June 1997.

[27] M. Donhahoo and E. Zegura, "Core Migration for Dynamic Multicast Routing,"ICCCN'96, September 1996.

[28] B. Waxman, "Routing of multipoint connections," IEEE Journal on Selected Areas in Communications, 6(9):1617-1622, 1988.

[29] D.Wall, "Mechanisms for broadcast and selective Broradcast,"June 1980,PhD Thesis, Stanford University.

[30] C. Shields, J.J. Garcia-Luna-Aceves, "The ordered core based tree protocol," INFOCOM 97.

[31] Andrew Tanenbaum, "Computer Networks", Prentice Hall.